The Graduate Institute of Science and Engineering

M.Sc. Thesis in Electrical and Computer Engineering

A STUDY ON THE COMPUTATIONAL COMPLEXITY REDUCTION OF PETRI NET BASED LIVENESS-ENFORCING SUPERVISORS IN FLEXIBLE MANUFACTURING SYSTEMS

by

Sunusi Garba MOHAMMED

June 2014
Kayseri, Turkey

A STUDY ON THE COMPUTATIONAL COMPLEXITY REDUCTION OF PETRI NET BASED LIVENESS-ENFORCING SUPERVISORS IN FLEXIBLE MANUFACTURING SYSTEMS

by

Sunusi Garba MOHAMMED

A thesis submitted to

The Graduate Institute of Science and Engineering

of

Melikşah University

In partial fulfillment of the requirements for the degree of

Master of Science

in

Electrical and Computer Engineering

June, 2014
Kayseri, Turkey

APPROVAL PAGE

This is to certify that I have read the thesis entitled "A Study on the Computational Complexity Reduction of Petri Net Based Liveness Enforcing Supervisors in Flexible Manufacturing Systems" by Sunusi Garba Mohammed and that in my opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science in Electrical and Computer Engineering, the Graduate Institute of Science and Engineering, Melikşah University.

June 6, 2014 Prof. Dr. Murat Uzam
Supervisor

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

June 6, 2014 Prof. Dr. Murat Uzam
Head of Department

Examining Committee Members

Title and Name		Approved
Prof. Dr. Murat Uzam	June 18, 2014	_____
Yrd.Doç. Dr. Ercan Şevkat	June 18, 2014	_____
Yrd.Doç. Dr. GökhanÖzgür	June 18, 2014	_____

It is approved that this thesis has been written in compliance with the formatting rules laid down by the Graduate Institute of Science and Engineering.

Prof. Dr. M. Halidun Keleştemur
Director

June 2014

A STUDYON THE COMPUTATIONAL COMPLEXITY REDUCTION OF PETRI NET BASED LIVENESS-ENFORCING SUPERVISORS IN FLEXIBLE MANUFACTURING SYSTEMS

Sunusi Garba MOHAMMED

M.Sc. Thesis in Electrical and Computer Engineering
June 2014

Supervisor: Prof. Dr. Murat UZAM

ABSTRACT

The existence of the shared resources in flexible manufacturing systems (FMS) may lead to deadlock conditions. In the context of FMS, deadlocks are undesirable in which the whole system is blocked. The most stimulating problems in FMS, design and operation are to apportion the shared resources to work perfectly without any deadlocks. Petri nets (PN) are considered as a general mathematical tool to handle deadlock problems.

The computational complexity of PN based liveness-enforcing supervisors of FMS is one of the current research topics studied in the related literature. In this work a method is proposed to improve the computational efficiency involving a PN model with a fixed structure.

Index Terms: Discrete event systems, Flexible manufacturing system (FMS), Deadlock prevention, Petri nets.

ESNEK ÜRETİM SİSTEMLERİNDE CANLILIK SAĞLAYICI GÖZETİCİLERİN HESAPLAMA KARMAŞIKLIĞININ AZALTILMASI ÜZERİNE BİR ÇALIŞMA

Sunusi Garba MOHAMMED

Yüksek LisansTezi– Elektrik ve Bilgisayar Mühendisliği
Haziran 2014

Supervisor: Prof. Dr. Murat UZAM

ÖZ

Esnek üretim sistemlerinde (Flexible Manufacturing Systems – FMS) paylaşılan kaynakların varlığı kördüğüm durumlarına sebep olabilir. FMS bağlamında, bütünsistem bloke olduğundan kördüğümler istenmez. FMS tasarımında ve işleyişinde en çok çalışılan problemler, herhangi bir kördüğüm olmaksızın paylaşılan kaynakları mükemmel bir şekilde taksim etmektir. Petri ağları (PN) kördüğüm problemlerinin üstesinden gelmek için kullanılan genel bir matematiksel araç olarak Kabul edilmektedir.

FMS'te Petri ağları tabanlı canlılık-uygulanması denetçilerin hesaplama karmaşıklığı ilgili literatürde incelenen güncel araştırma konularından biridir. Bu çalışmada, sabit yapılı bir Petri ağı modelinde hesaplama verimliliğini artırmak için bir yöntem önerilmiştir.

AnahtarKelimeler: Ayrık olay sistemleri, Esnek Üretim sistemi (FMS), Kördüğüm önleme, Petri ağları.

DEDICATION

I dedicated this thesis to my beloved governor of Kano state Nigeria, in person of his Excellency Dr. Rabiu Musa Kwankwaso.

ACKNOWLEDGEMENT

This work was supported by the research grant of the Scientific and Technological Research Council of Turkey (Türkiye Bilimsel ve Teknolojik Araştırma Kurumu - TÜBİTAK) under the project number TÜBİTAK-112M229. I would like to express my sincere appreciation to my supervisor Prof. Dr. Murat Uzam for his guidance and support throughout the course of my M.Sc. Program.

I wish to express my gratitude to my parents, family and friends for their words of encouragement and support during my M.Sc. Program.

TABLE OF CONTENS

LIST OF TABLES

TABLE

LIST OF FIGURES

FIGURE

LIST OF SYMBOLS AND ABBREVIATIONS

SYMBOL/ABBREVIATION

DES	Discrete event system
FMS	Flexible manufacturing system
PNM	Petri net model
RPNM	Reduced Petri net model
PI_S	Place invariants
FBM	First- met bad marking
RG	Reachability graph
DZ	Dead zone
LZ	Live zone
P	Place
t	Transition
M_0	Initial marking
C	Monitor/Control place
Eq	Equation
D_{PI}	Place invariant related incidence matrix
L_{PI}	Marked activity places
N_{RG}	The number of states in the reachabilty graph
N_{LZ}	The number of states in the live zone
N_{DZ}	The number of states in the dead zone
PN	Petri net
$\forall$	For all
$\exists$	There exist
$\nexists$	There not exist
$\cup$	Union

CHAPTER 1

INTRODUCTION

A discrete event system (*DES*) is a discrete state event-driven system in which its state of evolution and system behavior depends totally on the occurrence of asynchronous discrete events over time [1]. Varieties of systems, especially technological ones are discrete state systems. Applications of discrete event systems include, computer communication networks, monitoring and controlling systems of large buildings and automated manufacturing systems. Discrete event systems are commonly studied at two levels: logical and performance levels.

One of the promising tools used for describing and analyzing *DES* is Petri nets, a mathematical tool used by researchers and engineers for modeling and analyzing DES_S, characterized by their ability to represent operation sequences, concurrency, conflict, mutual exclusion, synchronization, and resource sharing in systems.

In Petri net formalism, liveness is an important property of system safety. Liveness implies the absence of global or local deadlock situations in a system [2].

A manufacturing system is a collection of manufacturing activities that are usually a transformation process by which raw materials, labor, energy and equipment are brought together to produce high quality products [3]. A manufacturing system consists of two major subsystems: a physical subsystem and a control subsystem. Specifically machine tools, robots, buffers, conveyors and automated guided vehicles can be used to represent a physical subsystem while the control subsystem also called the decision making subsystem, can be used to determine how to organize the physical subsystem in order to optimize the process. The activities in a manufacturing system can be seen as a sequence of discrete

events that are designed in line of the production requirements. More than one event can occur at the same time. In such a system, the time between events is usually different. The change of system states is driven by the occurrences of events; therefore it is possible for one event to trigger a series of events. The following represents the characteristics of *DES*:

1. Concurrency or parallelism: In a discrete event system many operations may take place at the same time, i.e., simultaneously.
2. Asynchronous operations: The evolution of system events occurs without a regular or predictable time relation to other events.
3. Event driven: The system behavior can be described by a discrete set, in which changes in states are caused by event occurrences. In other words, the effect of an event may propagate through the system.
4. Sequential relation: Some events in discrete event systems must occur in a sequential way.
5. Conflict: This may occur when two or more events require a common resource at the same time.
6. Non-determinism: Non-determinism is the result of conflicts, i.e. different evolutions may be possible from a given state.
7. Deadlock: It is a system state where none of the processes can continue. In manufacturing systems, the occurrences of deadlocks are undesirable situation often caused by improper resource-sharing. This situation is undesirable and is usually the result of system design; deadlock detection is useful in an automated manufacturing system design.

In a flexible manufacturing system (*FMS*), raw parts of different types enter the system at discrete points of time and are processes simultaneously, sharing a finite number of resources, such as machine tools, robots, buffers and vehicles. Every part has a particular operation flow that determines the trend in which resources must be provided to the part. The process in production sequences are said to be concurrent process, and they have to compete for the limited number of resources. Therefore, this competition can cause deadlocks. Digraphs, automata and Petri nets are three important mathematical tools to

handle deadlock problems in resource allocation systems, Petri nets are well suited to handle deadlock problems [4, 5].

1.1 DEADLOCK-HANDLING STRATEGIES

There are four strategies to handle deadlocks in automated manufacturing systems: deadlock ignoring, prevention, avoidance, and detection and recovery [5, 6].

1. Deadlock ignoring: It is employed in a resource allocation system if the probability of deadlock-control strategies is technically or financially difficult.
2. Deadlock prevention: It is considered to be a well-defined problem in resource allocation studies, usually an offline computational mechanism can be employed to control the request for resources to make sure that deadlocks never occur. Therefore, the goal of a deadlock prevention method is to impose constraints on a system's evolution to prevent it from reaching deadlock states.
3. Deadlock avoidance: Resources are allocated to a process provided the resulting state is safe. This implies that at least there exists one execution sequence that permits all processes to run to completion.
4. Deadlock detection and recovery: Although, resources are allocated to a process without monitoring, resource allocation and requests are monitored periodically to make sure whether a set of processes is deadlock. Deadlock detection algorithms can be used for this examination if a deadlock is obtained. Then, the system regains from it by stopping one or more deadlocked processes. The effectiveness of this approach relies on the response time for implementation algorithms for deadlock detection and recovery. In general, when several types of shared resources are considered, these algorithms demand a huge amount of data and may become complex.

Flexible manufacturing systems (*FMS*) have introduced tremendous improvements in system performance compared to classical manufacturing systems. *FMS* related proble-ms such as design, production planning and deadlock problems should be properly consider- ed before the installation. In [7] different approaches in deadlock handling strategies (deadlock ignoring, deadlock prevention, deadlock avoidance and

deadlock detection and recovery) are proposed to deal with problems of FMS_S at various stages. Deadlock problems in FMS_S have gained much attention from industry. Petri nets as a graphical and mathematical tool prove to be the most powerful method for the study of FMS_S.

1.1.1 LITERATURE REVIEW

Absence of deadlocks is critical in systems that are expected to operate in an automated way, they include life-support systems, nuclear plants, transportation control systems, and automated manufacturing systems. A systematic and efficient method to prevent, avoid, and detect deadlocks is of primary importance for them.

Over the past two decades, deadlock-control research received much attention from academic and industrial communities, leading to ample resolution methodologies, most of which were based on Petri nets. This section aims to present a literature review of deadlock control strategies for FMS with a focus on deadlock prevention [7].

A powerful feature of Petri nets is their ability to analyze good behavioral properties of the modeled systems such as deadlock freedom or liveness. Liveness is an important behavioral property of nets. It corresponds to the absence of global and local deadlock situations. A major breakthrough to evaluate the liveness of Petri nets and to synthesize liveness-enforcing Petri nets supervisors is the formal characterization of the non liveness of Petri nets through the formation of a particular structural object, which is known as siphons. Many deadlock prevention policies characterize the deadlock behavior of a system in terms of siphons and utilize this characterization to control or prevent deadlocks [8]. However, it is well known that the computation of minimal siphons is *NP*-complete. Consequently, it is very time-consuming or even impossible in the case of large-size systems. It is of significance if a deadlock prevention policy can avoid the complete siphon enumeration [9].

For a fixed net structure with an initial marking, once the liveness requirements are established, it is easy to decide its supervisor when the initial marking changes. It is shown that reachability graph-based approaches can usually find an optimal supervisor if it exists.

However, they suffer from expensive overhead, since the complete state enumeration of a Petri net is exponential with its size and initial marking. The deadlock-prevention policies that use partial reachability graphs seem to provide a tradeoff between computational cost and behavioral permissiveness. For a fixed net structure with a new initial marking, all the computation needs to be carried out afresh, since a reachability graph is sensitive to both net structure and initial marking. This is proved to be computationally inferior in comparison with siphon-based strategies, since siphons are pure structural objects whose computation is independent of initial markings.

Currently studied problems in Petri net based deadlock resolution in FMS are computational complexity, structural complexity, and behavioral permissiveness that are major criteria when designing a liveness-enforcing Petri net supervisor for a plant model [10].

Computational complexity results from the complete siphon or reachability graph enumeration that is necessary to compute a supervisor [11]. As known, the number of siphons grows fast and in the worst case grows exponentially with respect to the size of a net model.

Behavioral permissiveness problem is referred to as the fact that the permissive behavior of a plant net model is overly restricted by the deadlock prevention policy, i.e., the supervisor excludes some safe (admissible) states. This is so since the output arcs of a monitor are led to the source transitions of the net model, which limits the number of work pieces to be released into and processed by the system, a source transition is the output transition of an idle place, which models the entry of raw parts into the system.

In this thesis a new method is proposed for the synthesis of liveness-enforcing supervisors for small-sized Petri net models with a large initial marking of FMS suffering from deadlocks. The remainder of this thesis is organized as follows. Chapter 2 reviews the basic Petri net concepts related to this thesis. Chapter 3 presents the proposed synthesis method. Chapter 4 considers some examples to show the applicability of the proposed synthesis method. Conclusions are provided in chapter 5.

CHAPTER 2

BASIC CONCEPTS

In this Chapter firstly the definition of basic Petri nets and their properties are provided. Then Petri net reduction approach is recalled. Next an iterative synthesis approach used in this thesis is reviewed. Place invariant (*PI*) based computation of monitors (control places) is another topic considered in this Chapter. Finally a redundancy test which is used to find necessary monitors from a liveness-enforcing supervisor is recalled.

2.1 PETRI NETS

Petri nets as a mathematical tool have a number of properties. When interpreted in the context of modeled manufacturing system, these properties allow one to identify the presence or absence of the functional properties of the system. In this section, some definitions and concepts of Petri nets are provided [18].

A Petri net is a five-tuple, $PN = (P, T, F, W, M_0)$ (2.1)

Where:

- $P = \{ p_1, p_2, \ldots, p_m \}$ is a finite set of places, where $m > 0$. (Drawn as circle in the graphical representation).
- $T = \{ t_1, t_2, \ldots, t_n \}$ is a finite set of transitions, where $n > 0$, (drawn as bars or square boxes) with $P \cup T \neq \emptyset$, and $P \cap T \neq \emptyset$, $F \subseteq (P \times T) \cup (T \times P)$ is the set of all directed arcs, where $P \times T \rightarrow N$ is the input function that defines the set of directed arcs from P to T, and $T \times P \rightarrow N$ is the output function that defines the set of directed arcs from T to P, where $N = \{0, 1, 2, \ldots\}$.

- *W: F* → *N* is a mapping that assigns a weight to any arc.
- M_0*: P* → *N* is called a net system or marked net.

The set of input (resp., output) transitions of a place *p* is denoted by $^{\bullet}p$ (resp., $p^{\bullet}$). Similarly, the set of input (resp., output) places of a transition t is denoted by $^{\bullet}t$ (resp., $t^{\bullet}$) [7]. The graphical structure of a *PN* is a bipartite directed graph: the nodes belong to two different classes (places and transitions) and the edges (arcs) are allowed to connect only nodes of different classes (multiple arcs are possible in the definition of the input and output relations). Fig. 2.1 shows an example *PN*.

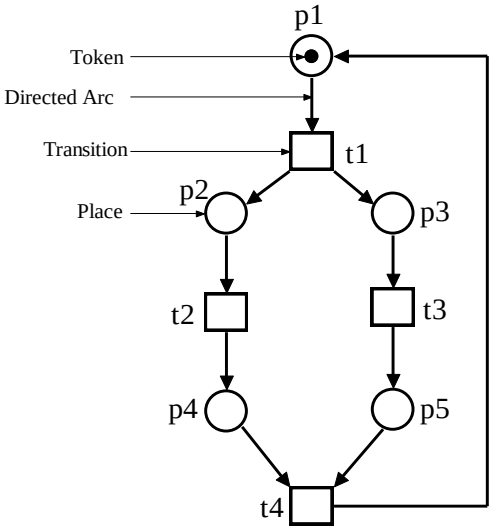

Figure 2.1. A *PN* graph with input and output functions.

2.1.1 FIRING OF A PETRI NET

The firing of a Petri net is shown in Fig. 2.2 where there are four places $P = \{p_1, p_2, p_3, p_4\}$ and three transitions $T = \{t_1, t_2, t_3\}$ as shown in Fig. 2.2.(a). Transition t_1 is enabled because $M(p_1) = 1$, $pre(p_1, t_1) = 1$ and transition t_2 and t_3 are not enabled, because $M(p_2) = 0$ and $pre(p_2, t_2) = 1$, $pre(p_2, t_3) = 1$. When transition t_1 fires, it removes one token from place p_1 and deposits one token in place p_2, as shown in Fig. 2.2.(b). In this case, transitions t_2 and t_3 become enabled because $M(p_2) = 1$ and $pre(p_2, t_2) = 1$ or $M(p_2) = 1$ and $pre(p_2, t_3) =$

1. When transition t_2 fires, it removes one token from place p_2 and deposits one token in place p_3 as shown in Fig. 2.2.(c) also when transition t_3 fires, it removes one token from place p_2 and deposits one token in place p_4 as shown in Fig. 2.2.(d).

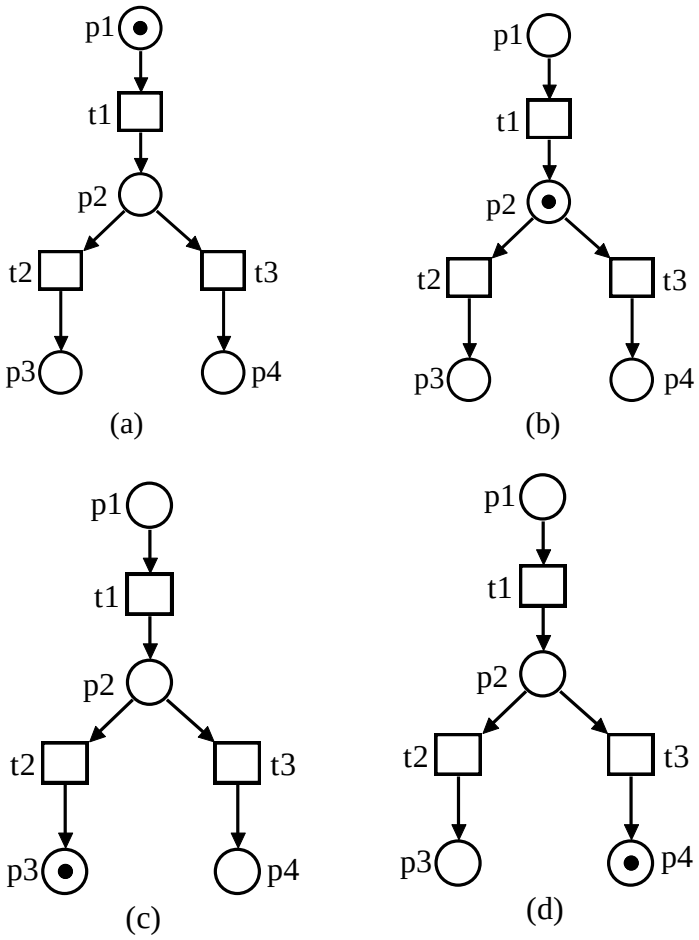

Figure 2.2. A Petri net with: (a) Initial marking. (b) Marking after t_1 fires. (c) Marking after t_2 fires. (d) Marking after t_3 fires.

The places, tokens and transitions must be assigned a meaning for proper modeling of systems. In general, places identify the conditions of the parts of the system (working, idle, queuing, and failed). The presence of one or more tokens in a place represents the availability of a particular resource or presence of a condition being met. Transitions describe the passage from one condition to another (end of a task, failure, and repair). An

event occurs (a transition fires) when all the conditions are satisfied (input places are marked) and give concession to the event.

2.1.2 PROPERTIES OF PETRI NETS

There are two types of properties, namely: behavioral and structural properties. The former depends on the initial marking of the Petri net, while the latter does not depend on the initial marking. In this section, the focus is on some basic behavioral properties such as reachability, boundedness, liveness, safeness, and conservativeness.

Reachability: The firing of an enabled transition will change the token distribution (marking) of a Petri net. A marking M_n is said to be reachable from an initial marking M_0 if there exists a sequence of firings that transforms M_0 to M_n. A firing or occurrence sequence is denoted by $\sigma = t_1, t_2, t_3, t_4 \ldots, t_n$ in this case M_n is reachable from M_0 by σ the following condition is used: $M_0[\sigma > M_n$.

- Boundedness: A Petri net is said to be k-bounded or simply bounded if the number of tokens in each place does not exceed a finite number k for any marking reachable from the initial marking M_0.
- Liveness: A transition is potentially firable in M if there exists a sequence of transition firings which leads to a marking in which the transition is enabled.
- Safeness: A place is safe if the token count does not exceed one in any marking of $R(M_1)$. A PN is safe if each place is safe.
- Conservativeness: A PN is strictly conservative if the total number of tokens in all of its places for all reachable markings is constant [12].

Important concepts such as liveness, boundedness, and proper termination can be defined by using Petri nets. After modeling a system with a Petri net, many fascinating properties of the system can be shown by analyzing the Petri net. A Petri net prone to deadlocks can be discovered by verifying the liveness of the net. However, the complexity of the model is drastically increased with the number of states in the reachability graph of

the Petri net. This complexity often gives rise to substantial errors on modeling and analyzing the system [13].

2.1.3 PETRI NET REDUCTION APPROACH

The analysis of a large-scale Petri net is faced with the state explosion problem, reduction of system size is a general approach to maintain the complexity in the field of system engineering and large scale systems [14], [15]. Our basic principle in maintaining the complexity is to reduce the size of reachability graph, by a Petri net reduction method.

Petri net reduction is a procedure that transforms Petri nets into their reduced nets while maintaining some desirable properties of the original nets. This technique reduces the number of states in a reachability graph. As a result, the analysis of the simplified net can provide sufficient information for the understanding of the original net. Moreover, the verification and validation of the modeled system can be successfully achieved on the reduced models.

Petri net reduction approach is a well-known method to derive the properties of a complex Petri net model, while preserving the concerned properties, such as boundedness, liveness and reversibility [12]. It is possible to analyse and derive the properties of a complex Petri net model, by simplifying the subnet or structure. In this section, some simple reduction rules are considered. A set of easy-to-use reduction rules is given in Fig. 2.3, including the following:

Rule 1: Fusion of series places as shown in Fig. 2.3.(a)

Rule 2: Fusion of series transitions as shown in Fig. 2.3.(b).

Rule 3: Fusion of parallel places as shown in Fig. 2.3.(c).

Rule 4: Fusion of parallel transitions as shown in Fig. 2.3.(d).

Rule 5: Elimination of self-loop places as shown in Fig. 2.3.(e).

Rule 6: Elimination of self-loop transitions as shown in Fig. 2.3.(f).

It can be proven that these six operations preserve the properties of liveness, safeness and boundedness, when they are applied to reduce a Petri net. That is, let (N, M_0)

and (N', M'_0) be the Petri nets before and after one of the above mentioned operations. Then (N', M'_0) is live, safe, or bounded if (N, M_0) is live, safe, or bounded, respectively [12]. There exist many transformation techniques for Petri nets. For other complicated reduction rules and their applications the reader is referred to [16]. An application of reduction approach to analysis of a Petri net model for the etching area of an *IC* fabrication system can be found in [17].

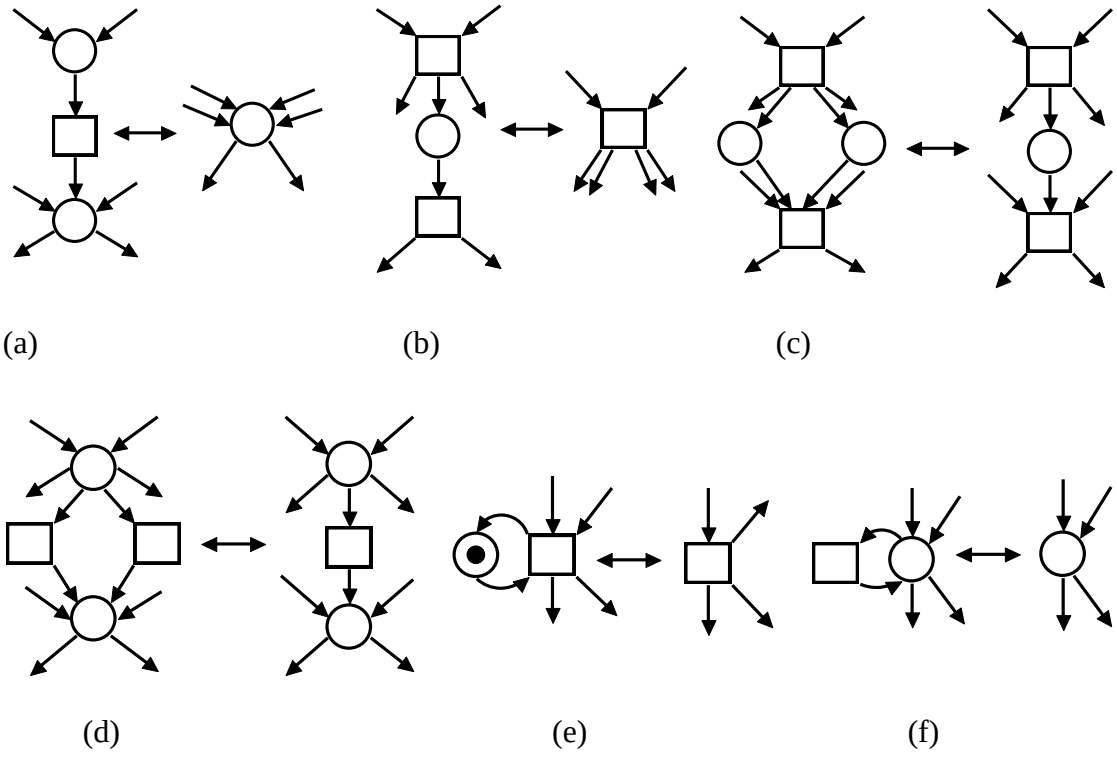

Figure 2.3. Set of easy-to-use Petri net reduction rules, preserving liveness, safeness and reversibility.

2.1.4 AN ITERATIVE SYNTHESIS APPROACH

In this thesis the synthesis of monitors (control places) is carried out based on the method proposed in [18]. This section recalls the Petri net-based deadlock-prevention policy reported in [18]. In this policy two improvements were proposed over the previous method [19]. The first is the use of the Petri net reduction approach to simplify the Petri net model of an *FMS*, and the second is the use of simplified controller computation. The improved deadlock-prevention policy proposed can be used for large FMS_S, requiring complex Petri net models. It is well known that as Petri nets become larger, the *RG* of the Petri nets grows exponentially with the size of the Petri net. This problem is called the 'state explosion problem'. Due to this problem for analysis of complex Petri net models, we use the Petri net reduction approach to simplify very large PNM_S so as to perform necessary computations easily in order to obtain liveness enforcing supervisors for FMS_S. The aim is to obtain live, i.e. deadlock-free, controlled PNM_S of FMS_S. A control policy was defined as the addition of new constraints to the system such that its initial behaviour is restricted to a set of states that are considered as 'good states', which allow the system to evolve without reaching a deadlock state. When doing this, it is also made sure that all possible good states of the system can still be reached under the control policy. It is important to note that the reduced Petri net models (*RPNM*) are only used for necessary computations to obtain the liveness enforcing supervisors by reducing the original *PNM*. Due to the fact that by using the Petri net reduction approach the liveness property of the original (uncontrolled) *PNM* is preserved, when the synthesized new net elements are added into the initial (uncontrolled) *PNM* a live controlled *PNM* of the system is also obtained.

In this iterative deadlock-prevention approach, the *RG* of the *PNM* of an *FMS* is the starting point for the definition of a control policy. The *RG* of the *PNM* is split into a deadlock-zone (*DZ*) and a live-zone (*LZ*). The former may contain deadlock states (markings), partial deadlock states, and states which are inevitable lead to deadlocks or livelocks. The latter constitutes remaining good states of the *RG* representing the optimal system behaviour. The control policy is based on the exclusion of the *DZ* from the RG, while trying to make sure that every state within the *LZ* may still be reached. When

studying the relationship among the markings of a *DZ* of a *PNM*, it is succeeded to unveil the fact that it is possible to obtain a live model by means of a set of what is called first-met bad markings. A first-met bad marking (*FBM*) is defined as a marking residing within the *DZ*, and it represents the very first entry from *LZ* to *DZ*. In fact, an *FBM* does not have to be a deadlock state. However, it is not possible to reach the initial marking from an *FBM* and an *FBM* may lead the way to a deadlock state or a group of livelocked states. There may be three groups of places in a *PNM* of an *FMS*: resource places, activity places, and sink/source places. Resource places represent either shared or non-shared resources, and initially there are tokens in these places representing the number of available resources. Activity places represent an action to process a part in a production sequence by a resource (machine, robot, etc.), and initially there are no tokens in these places, Initially, tokens put into sink/source places represent the number of concurrent activities which can take place in a production sequence. In some models, it may be possible not to use them. In cyclic models, a sink place is also a source place and vice versa. From an *FBM*, only the markings of activity places are considered. The number of tokens in the marked subset of the activity places represents the first entry into *DZ*. Then, the objective is to prevent the marking of the subset of the activity places of the *FBM* from being reached. Therefore, the marking of the subset of the activity places is characterized as a *PI* of the *PNM*. In the PI relating to an *FBM*, the sum of tokens within the subset of the activity places has to be at most one token less than their current value within the *FBM* in order not to reach the *FBM*. A *PI* can be implemented by a liveness enforcing supervisor (a control place with its related arcs and initial marking) the simplified version of the method proposed in [20] is used in order to obtain a control place (also called ‘a monitor’) from a *PI* as described. Note that, in some models, *DZ* is made up of livelocks without a deadlock state. In such a case, the objective is still the same: to make the model live. The deadlock-prevention method proposed in this thesis makes use of the following iterative approach, and the following shows the design procedure.

Algorithm 1: **An iterative synthesis method for liveness-enforcing supervisors of flexible manufacturing systems** [18].

Input: The reduced Petri net model (*RPNM*) of *FMS* prone to deadlocks.

Output: A live controlled *RPNM* for the *FMS*, *RPNMC* for short.

1. Compute the reachability graph, RG_{i+1}, i = 0, 1, 2, … of the reduced Petri net model $RPNMC_{i+1}$, i = 0, 1, 2, … ($RPNMC_i$ = *RPNM* for the first iteration).
2. If the RG_{i+1}, i = 0, 1, 2, …, is live, then go to step 7.
3. Find the first-met bad marking ($FBM_{i+1,}$ i = 0, 1, 2, …) within the RG_{i+1}, i = 0, 1, 2, … taking the system evolution from LZ_{i+1} to DZ_{i+1} , i = 0, 1, 2, … .
4. Define a *PI* ($PI_{i+1,}$ i = 0, 1, 2, …) from the subset of the marked activity places of the $FBM_{i+1,}$ i = 0, 1, 2, … .
5. Compute a liveness enforcing supervisor (a control place) C_{i+1}, i = 0, 1, 2, … , for the place invariant PI_{i+1}, i = 0, 1, 2, … using the simplified invariant-based control method.
6. Add the computed liveness enforcing supervisor (control place) $RPNMC_i$, i = 0, 1, 2, … , into the reduced Petri net model ($RPNMC_{i+1}$: = $C_{i+1,}$ + $C_{i+1,}$ i = 0, 1, 2, … $RPNMC_i$,= RPNM for the first iteration) and go to step 1.
7. Add all computed liveness enforcing supervisors to the reduced Petri net model (*RPNM*) and thus obtain the live controlled *RPNM* for the *FMS*.

End of Algorithm 1.

The above method is easy to use, very effective, and straightforward. It has been tested against all the significant Petri net models prone to deadlock currently available in the literature with success. The reachability graph analysis of *PNMS* can be carried out by currently available Petri net analysis tools. In this thesis, PNTOOLS [22] is used, in which

the *DZ* of a given Petri net model is provided. From DZ_{i+1} , i = 0, 1, 2, … ., the first entry ispicked up as $FBM_{i+1,}$ i = 0, 1, 2, … .

Marking m and m' are said to equally mark a set of places, *S*, if and only if $\forall\ p \in S$, m(p)=m(p').

Lemma 1: At any *FBM*, at least two activity places are marked.

This is because in any PN model of an *FMS*, it is impossible to have a deadlock if only one activity is being executed (implying only one activity place is marked). The circular wait is not possible.

Theorem 1: The Algorithm 1 terminates in a finite number of steps for a bounded *RPNM* of an *FMS*.

Proof: We need to show that at each iteration, the number of bad states is reduced and no new bad states are generated. Given $PNMC_i$, at the *i*th iteration, we add a *P*-invariant to it resulting in $RPNMC_{i+1}$. The *P*-invariant is marked with the number of tokens less than the sum of tokens in all marked activity places at the selected *FBM*. Due to Lemma 1, this number is greater than zero. Under these *P*-invariant constraints, it is no longer possible to reach the selected *FBM* and any other markings that can mark equally the activity places at the *FBM* in $RPNMC_i$. Furthermore, it adds no new nodes in the *RG* (thus no new *FBMs*). Since the number of *FBMs* is limited for a bounded *PN*, the algorithm will terminate.

An *FBM* is optimal if and only if none of the good markings in the *RG* marks equally the activity places at this *FBM*. The condition is called the optimality condition.

Theorem 2. If at each iteration, an optimal *FBM* exists and is selected, then the algorithm leads to the live *PNMC* that is maximally permissive.

Proof: The Algorithm 1 will then eliminate no good markings at each iteration when a new *P*-invariant is added based on the optimal *FBM*. Thus all good markings will be preserved, leading to the optimally controlled *PNM* that is live. Unfortunately, the optimality condition cannot be met at each iteration for some PNM of an FMS as illustrated in [21].

In other words, invariant based control theory cannot guarantee the optimality in general given *FMS* and its *PNM* [18].

2.1.5 PI BASED MONITOR COMPUTATION

In this section, the controller computation method proposed of [20] is reviewed. In [20], a computationally efficient method was presented for constructing Petri net controller for a discrete event system modeled by a Petri net. The controller consists of places and input-output arcs, and is computed based on the concept of Petri net place invariants and is able to enforce logical and algebraic constraints containing elements of the marking and firing vectors. The system (also known as the plant or the process net) to be controlled is modeled by a Petri net with n places and m transitions. The incidence matrix of the plant net is D_P. The controller net is a Petri net with incidence matrix D_C made up of the transitions of the plant net and a separate set of places. The controlled plant is the Petri net with incidence matrix D made up of both the original plant net and the added controller. The control goal is to force the plant to obey constraints of the form:

$$\sum_{i=1}^{n} l_i \mu_i \leq \beta \tag{2.2}$$

where μ_i is the initial marking of place p_i, and the l_i and β are integer constants. By introducing a non negative slack variable μ_c this inequality constraint can be transformed into an equality as follows:

$$\sum_{i=1}^{n} l_i \mu_i + \mu_c = \beta \tag{2.3}$$

In this case, the slack variable denotes a new place p_c, generally called a control place or a monitor, which holds the extra tokens required to meet the equality. The control place ensures that the weighted sum of tokens in the places of the plant net is always less than or equal to β. The controller net, composed of the control places and their input and output arcs, maintains the inequality constraint. Place invariants are sets of place whose

token count remains constant for all possible markings. All constraints of type (1) can be grouped in matrix form as follows:

$$L \,.\, \mu_p \leq b \tag{2.4}$$

where μ_p is the marking vector of the plant Petri net model, L is an $n_c \times n$ integer matrix, b is an $n_c \times 1$ integer vector and n_c is the number of the constraints of type (1). All place invariants of type (2) can be grouped in the matrix form as follows:

$$L \,.\, \mu_p + \mu_c = b \tag{2.5}$$

where μ_c is an $n_c \times 1$ integer vector, representing the marking of the control places. Finally, given a plant Petri net model D_P and the constraints the plant must satisfy, namely L and b, the Petri net controller D_C is defined as follows:

$$D_C = -L \,.\, D_P \tag{2.6}$$

The initial marking of the controller Petri net μ_{c0}, is calculated in such a way that the place invariant Eq. (2.5) is initially satisfied. Therefore the initial marking vector is as follows:

$$\mu_{c0} = b - L\, \mu_{p0} \tag{2.7}$$

In [20], it was assumed that all place invariants to be enforced on the plant net are given and therefore in the controller computation the incidence matrix D_P of the plant net is used. As a result the controller net can be computed with one matrix multiplication as shown in Eq. (2.6). In the iterative deadlock prevention approach we need to compute one control place with its input-output arcs at each iteration. In the previous work [23], the method of [20] is used, namely Eqs. (2.6) and (2.7) as it is, for the computation of the controller at each iteration, the incidence matrix D_P of the plant plus the controller net obtained in the previous iterations is used. This means that for the computation of the controller the matrix multiplication must be performed with a very big incidence matrix at each iteration. However, it can be observed that the control net, i.e. the control place and its input-output arcs, to be computed at each iteration within the deadlock prevention approach

consists of only the input-output transitions of those places which appear within the place invariant.

This means that for computing one control net for a single place invariant there is no need to use the incidence matrix D_P of the plant net. Rather, this computation can be carried out by using the incidence matrix D_{PI} of the place invariant related net. Of course at each iteration, for each controller computation a different incidence matrix D_{PI} must be used.

However, when we are dealing with very complex Petri net models this simplification is justified because otherwise we must use a very big incidence matrix D_P of the plant net at each iteration. Given a place invariant related net, i.e., a set of places with their input-output arcs and the constraint, the place invariant related net must satisfy, i.e., L_{PI} and b, we simplify (2.6) and as a result the Petri net controller D_C is defined as follows:

$$D_C = -L_{PI} . D \tag{2.8}$$

where D_{PI} is the incidence matrix of the place invariant related net with j places and k transitions, L_{PI} is a $j \times 1$ integer row vector representing the invariant related places and D_C is a $k \times 1$ integer row vector representing the incidence matrix of the computed controller net. The initial marking of the computed controller net μ_{c0}, can be found by:

$$\mu_{c0} = b - L_{PI} . \mu_{PI0} \tag{2.9}$$

where μ_{PI0} is the initial markings of the place invariant related places. Note that by using the (2.8) and (2.9) it is possible to simplify the computation of a Petri net controller when there is only one place invariant to be enforced on a plant Petri net. In the special case of (2.9), within our deadlock prevention approach a place invariant consists of only activity places. By definition, there is no token within the activity places initially. This means that $L_{PI} . \mu_{PI0} = 0$. As a result in our deadlock prevention approach, (2.9) becomes:

$$\mu_{c0} = b \tag{2.10}$$

This shows that when we obtain a place invariant to be enforced on a plant net in our deadlock prevention approach, the initial marking of the controller is equal to *b*. This also further simplifies the controller computation [21].

2.1.6 REDUNDANCY TEST FOR LIVENESS-ENFORCING SUPERVISORS OF FMS

In the previous section a method from [18] is recalled which is used for the computation of monitors (control places) in this thesis. It is well known that there may be some redundant monitors among the computed monitors. Therefore it is important to check whether or not the computed monitors are necessary. In this section a redundancy check method proposed for computed monitors is recalled from [24]. In Petri-net-based deadlock-prevention/liveness enforcing approaches, an *FMS* is modeled as a Petri net, and then the liveness enforcing supervisor (*LES*), consisting of a number of control places (*CPs*), together with their related arcs and initial markings, is computed as a Petri net. There may exist redundant *CPs* in a live Petri net (*LPN*) model, denoted by a net system (N_0, M_0), controlled by *n CPs*: $CP = \{C_1, C_2, \ldots, C_n\}$. In this paper, a *CP* is called redundant if removing it still keeps the net live. It should be noted that this definition is different from that of a redundant place in literature. Removing the latter does not change the net's reachability graph. Also, redundant *CPs* are not necessarily unique given a set of *CPs* used to make a deadlock-prone net live.

Redundancy Test Algorithm: Redundancy Test for *LES* of *FMS*.

Input: A live Petri net (*LPN*) model, denoted by a net system (N_0, M_0), of an *FMS*,

controlled by *n CPs*, $CP = \{C_1, C_2, \ldots, C_n\}$.

1. (Define) β_0: The number of reachable markings or states of reachability graph (R_0) of (N_0, M_0).

(Defined for Algorithm *A*) β_A: The number of reachable markings or states of R_A of (N_A, M_A), $n = j + k$, where n: The number of *CPs* of *LPN*, j: the number of redundant *CPs*, k: The number of necessary *CPs*.

(Defined for Algorithm *B*) β_B: The number of reachable markings or states of R_B of (N_B,M_B), $n = l + m$, where n: the number of *CPs* of *LPN*, l: the number of redundant *CPs*, m: the number of necessary *CPs*.

1. Apply Algorithm *A* to (N_0, M_0) and the resultant net system is denoted as (N_A, M_A).
2. Apply Algorithm *B* to (N_0, M_0) and the resultant net system is denoted as (N_B, M_B).

Output: If ($j > 0$) [for Algorithm *A*]

Then Output *A* = An *LPN*, denoted by a net system (N_A, M_A), controlled by k necessary *CPs*, there are j redundant *CPs*.

If $\beta_A = \beta_0$ then the controlled behaviour of (N_A, M_A) is the same as (N_0, M_0).

If $\beta_A > \beta_0$ then the controlled behaviour of (N_A, M_A) is more permissive than (N_0, M_0).

Else there is no redundant *CPs* obtained due to Algorithm A and therefore for Algorithm *A*:

Output = Input.

If ($l > 0$) [for Algorithm *B*]

Then Output *B* = an *LPN*, denoted by a net system (N_B, M_B), controlled by m necessary *CPs*, there are *l* redundant *CPs*.

If $\beta_B = \beta_0$ then the controlled behaviour of (N_B, M_B) is the same as (N_0, M_0).

If $\beta_B > \beta_0$ then the controlled behaviour of (N_B, M_B) is more permissive than (N_0, M_0).

Else there is no redundant *CPs* obtained due to Algorithm *B* and therefore for Algorithm *B*: Output = Input.

End of Redundancy Test Algorithm.

Algorithm *A*: Front-to-Back (*FTB*) redundancy test for *LES* of *FMS*.

Input: A live Petri net (*LPN*) model, denoted by a net system (N_0, M_0), of an *FMS*, controlled by *n CPs*, $CP = \{C_1, C_2, \ldots, C_n\}$.

1. (Initialize) $N_A = N_0$, $M_A = M_0$, $i = 1$, $j = 0$, $k = 0$.
2. Remove C_i from (N_A, M_A), denote the resultant net system by (N_i, M_i).
3. Check the liveness property of (N_i, M_i), compute the reachability graph (R_i) of (N_i, M_i) and define β_{Ai}, i.e., the number of reachable markings of R_i, If (N_i, M_i) is NOT LIVE.

Then put C_i back into (N_i, M_i), $k = k + 1$, which means that C_i is necessary to keep the *PN* model live.

Else [i.e., If (N_i, M_i) is LIVE], $j = j + 1$, which means that C_i is redundant.

If $\beta_{Ai} = \beta_0$ then the controlled behaviour of (N_i, M_i) is the same as (N_0, M_0).

If $\beta_{A\,i} > \beta_0$ then the controlled behaviour of (N_i, M_i) is more permissive than (N_0, M_0).

End if,

1. $N_A = N_{i,}$ $M_A = M_{i.}$
2. $i = i + 1$.
3. If $i \leq n$ then go to step 2.

Output: If $(j > 0)$.

Then Output = An *LPN*, denoted by a net system (N_A, M_A), controlled by k necessary *CPs*, there are

j redundant *CPs*.

If $\beta_A = \beta_0$ then the controlled behaviour of (N_A, M_A) is the same as (N_0, M_0).

If $\beta_A > \beta_0$ then the controlled behaviour of (N_A, M_A) is more permissive than (N_0, M_0).

Else there is no redundant *CPs* and therefore Output = Input.

End of Algorithm *A*.

Algorithm B: Back-to-Front (*BTF*) redundancy test for *LES* of *FMS*.

Input: *A* live Petri net model (*LPN*), denoted by a net system (N_0, M_0), of an *FMS*, controlled by *n CPs*, $CP = \{C_1, C_2, \ldots, C_n\}$.

1. (Initialize) $N_B = N_0$, $M_B = M_0$, $i = n$, $l = 0$, $m = 0$.
2. Remove C_i from (N_B, M_B), denote the resultant net system by (N_i, M_i).
3. Check the liveness property of (N_i, M_i), compute the reachability graph (R_i) of (N_i, M_i) and define β_{Ai}, i.e., the number of reachable markings of R_i, If (N_i, M_i) is NOT LIVE.

Then put C_i back into (N_i, M_i), $m = m + 1$, which means that C_i is necessary to keep the *PN* model live.

Else (i.e., If (N_i, M_i) is LIVE), $l = l + 1$, which means that C_i is redundant.

If $\beta_{Bi} = \beta_0$ then the controlled behaviour of (N_i, M_i) is the same as (N_0, M_0).

If $\beta_{Bi} > \beta_0$ then the controlled behaviour of (N_i, M_i) is more permissive than (N_0, M_0).

End if,

1. $N_B = N_{i,}$ $M_B = M_{i.}$
2. $i = i - 1$.
3. If $i \neq 0$ then go to step 2.

Output: If ($l > 0$).

Then Output = An *LPN*, denoted by a net system (N_B, M_B), controlled by m necessary *CPs*, there are *l* redundant *CPs*.

If $\beta_B = \beta_0$ then the controlled behaviour of (N_B, M_B) is the same as (N_0, M_0).

If $\beta_B > \beta_0$ then the controlled behaviour of (N_B, M_B) is more permissive than (N_0, M_0).

Else there is no redundant *CPs* and therefore Output = Input.

End of Algorithm *B*.

The Redundancy Test Algorithm makes use of both Algorithms *A* and *B*. The former tests each *CP* starting from number 1 to the end, i.e., to *n*, while the latter tests each

CP starting from number n to 1. Both tests may produce the same result or it may be possible to obtain different outcomes. It depends on the controlled live net system (N_0, M_0) considered. Of course if there is no redundant *CP* in an *LPN*, then the Algorithm Redundancy Test finds no redundant *CP*. In the existence of one or more redundant *CP* in an *LPN*, we may obtain the following results:

1. We may obtain the same set of redundant *CPs* and necessary *CPs*. In this case, the live behaviour of the Petri net model, controlled by the set of necessary *CPs*, may be the same as or more permissive than the original controlled net system, obtained with a smaller number of *CPs*.
2. We may obtain two different sets of redundant *CPs* and necessary *CPs*. The live behaviour of the Petri net model obtained with each set of necessary *CPs*, may be the same as or more permissive than the original controlled net system, obtained with a smaller number of *CPs*.

The Redundancy Test Algorithm is easy to use, very effective and straight forward. Its complexity is however, exponential with respect to the net size since it requires generating the reachability graph. At the worst cases, Algorithm *A* and Algorithm *B*, i.e. *BTF* and *FTB* redundancy tests respectively, also exhibit the same exponential complexity. When dealing with a particular case, their performance may vary significantly. The Redundancy Test Algorithm is applicable to any *LPN* consisting of a *PNM*, prone to deadlock, of an *FMS*, controlled by means of a set of *CPs*. It has been applied to a number of *LPN* currently available within the Petri net based deadlock prevention/liveness enforcing literature with success [24]. The liveness property can be checked and the reachability analysis can be carried out by currently available Petri net analysis tools. In this thesis, PNTOOLS [22] is used.

CHAPTER 3

SYNTHESIS OF LIVENESS-ENFORCING SUPERVISORS FOR FMS

In the synthesis of liveness-enforcing supervisors for *FMS*, in general the necessary computation is either expensive or impossible especially when dealing with either a large sized Petri net model or a small-sized Petri net model with a large initial marking. The problem is even worse when *RG* based methods are deployed. In this chapter a new and very effective method is proposed for the synthesis of liveness-enforcing supervisors for small-sized Petri net models with a large initial marking of *FMS* suffering from deadlocks. It is important to note that the computations of such supervisors are not possible with the currently available methods in the literature. The main idea of the proposed method is to identify shared resource places whose initial token values are too large. Then the token values of these special shared resource places are iteratively increased starting from only one token. For each instance a set of monitors (control places) are computed that make the related model live. Based on the place invariants (PI_S) of the computed monitors, generalized PI_S are obtained. The computation is carried out until generalized PI_S are valid for general case. After that, generalized PI_S can be used for all instances of the considered *PNM*. The following algorithm shows the proposed method.

Algorithm 2 : Computational Cost Reduction.

Input: The Petri net model (*PNM*) of a flexible manufacturing system (*FMS*) prone to deadlocks.

Output: The set of general *PIs* to force liveness on the given *PNM* in general case.

1. Obtain the reduced *PNM* (*RPNM*) by using Petri net reduction approach.
2. Define the set of adjustable shared resources whose token values are adjustable

3. For ($j = 1; j \leq 3; j$ ++)

{

3.j.1. Set the number of tokens in each adjustable shared resource as '*j*'.

3.j.2. Use Algorithm 1 and compute the set of monitors (control places) and their related place invariants (*PIs*) and carry out the redundancy test to find out necessary monitors.

3.j.3. Generalize the PIs computed in the previous step in such a way that the right hand side of the operator "≤" is defined based on the "token values of shared resources – 1".

}

4. List the generalized *PIs* for the general case that is applicable to all possibilities for the given PNM with adjustable shared resources.
5. Verify the correctness of the generalized *PIs* by computing the related monitors and by including them in the uncontrolled *PNM*.

End of Algorithm 2.

3.1 ILLUSTRATIVE EXAMPLE

In this section an example is considered to show the applicability of the proposed method. Fig. 3.1 shows the Petri net model (*PNM*) of an *FMS* for the two production sequences. Initially, it is assumed that there are no parts in the system. In the *PNM*, there are fourteen places, $P = \{p2 - p5, p7 - p10, p11 - p14, p21 - p22\}$ and ten transitions, $T = \{t1 - t10\}$. Places can be considered as the collection of six activity places $P_A = \{p5 - p10\}$ four resource places $P_R = \{p11 - p14\}$ and two sink/source places $P_{S/S} = \{p21, p22\}$. Places $p5 - p2$ represents the operation of shared resources for the part type $P1$. Similarly, Places $p7 - p10$ represents the operation of shared resources for the part type $P2$. The number of tokens in places $p21$, i.e., $\mu_{21} = 20$ and $p22$, i.e., $\mu_{22} = 20$, represent the number of concurrent activities that can take place for part types *P1* and *P2* respectively. Place *p11*

(*p12*, *p13* and *p14* respectively) denotes the shared resources. Initial markings of places *p11*, *p12*, *p13* and *p14* are all 5 as shared resources can process five parts at a time.

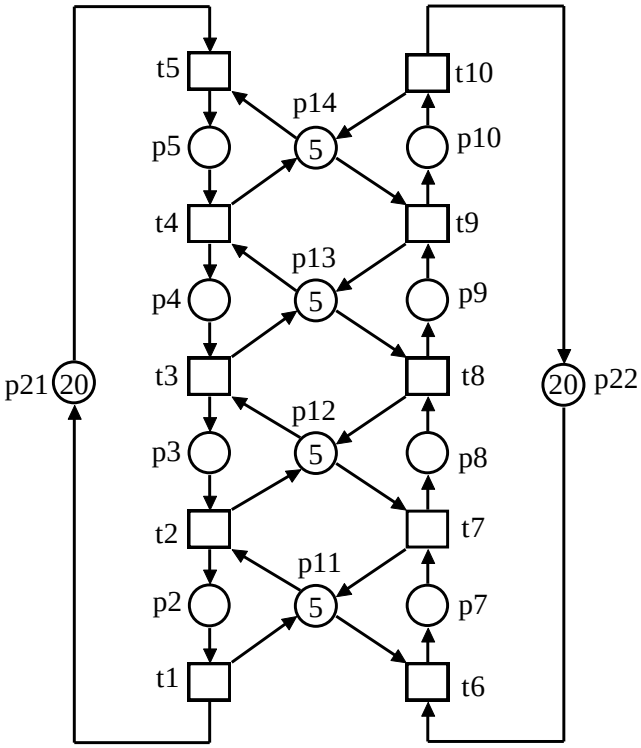

Figure 3.1. Petri net model (*PNM*) of an *FMS* for two production sequences.

It can be verified that the uncontrolled *PNM* shown in Fig. 3.1 is prone to deadlocks. It is necessary to deploy some control mechanisms to prevent deadlocks in it. There are 192975 states within the reachability graph (*RG*) of the uncontrolled *PNM*. The *DZ* contains 1406 bad states. Therefore, an optimal control policy should provide the live behavior (i.e. *LZ*) including 191569 states. Let us now apply the proposed method to this problem.

Step 1. Obtain the reduced *PNM* (*RPNM*)

In order to obtain the reduced *PNM*, sink/source places, namely *p21* and *p22* of Fig. 3.1 are removed together with their input/output arcs. It can be verified that the removal of

the places does not affect the behavior of the *PNM*. Both production sequences can be carried out as described before. When we remove them together with their input/output arcs, the *PNM* allows the same behavior. In this case, properties such as liveness, safeness and reversibility are preserved. Second by using Petri net reduction rules, series transition *t2* and *t1* (*t9* and *t10* respectively) can be merged as *t2* (*t9* respectively) [18]. We obtain the reduced *PNM* as shown in Fig. 3.2.

Step 2. Define the set of adjustable shared resources.

The adjustable shared resources are defined as *p11, p12, p13* and *p14*.

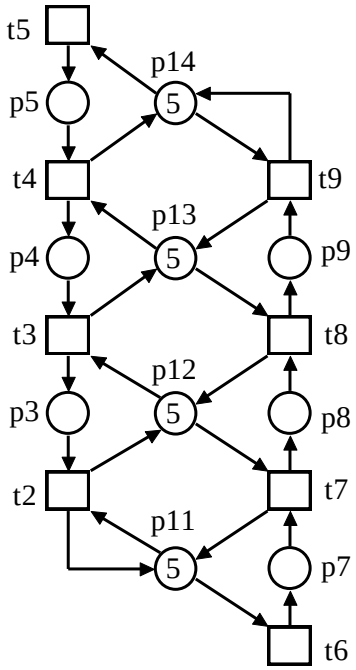

Figure 3.2. The Reduced *PNM* (*RPNM*).

Step 3.

Step 3.1.1 The number of tokens in the adjustable shared resource places are set to 1 ($M_0(p11) = M_0(p12) = M_0(p13) = M_0(p14) = 1$) as shown in Fig. 3.3.

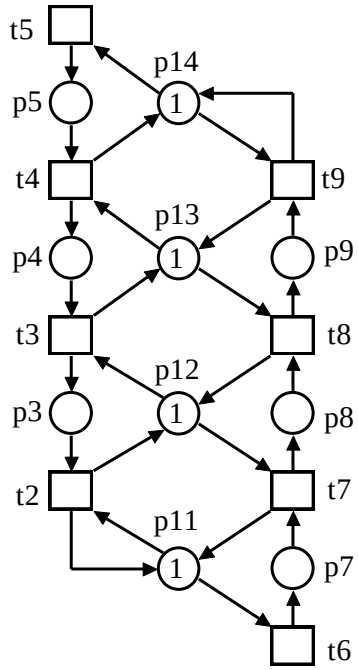

Figure 3.3. Reduced *PNM1* (*RPNM1*) with one token deposited in each shared resource place.

It is verified that the *RPNM*1 shown in Fig 3.3 suffers from deadlocks.

Step 3.1.2 Algorithm 1 is used to compute monitors and the redundancy test is used to find the necessary monitors.

When the *RPNM*1 shown in Fig. 3.3 is analyzed using PNTOOLS [22], it is seen that there are 32 states (*#S* in *RG*) in the RG_1 of the $RPNM_1$. For the liveness property, the following is obtained:

```
The liveness: The net is not live.
The following nodes (i.e states) are not full: 5, 9, 10, 14, 15,
16, 17, 18, 21, 22, 23, 24, 27, 28, 29, 31, 32.
```

These 17 bad states (*#S* in *DZ*) constitute the DZ_1 as indicated with red colour in Fig. 3.4 and the uncoloured portion shown in Fig. 3.4 are the remaining 15 good states (*#S* in *LZ*) represent the LZ_1. Therefore, an optimal deadlock-prevention policy should provide an answer to this problem such that 17 bad states are removed from the RG_1 and the remaining 15 good states are always reachable. Therefore the objective is to eliminate the

DZ_1. In the first iteration (i = 0) from the DZ_1, the first marking namely $FBM_1 = m_5$ provided in Table 3.1, is choosen as an *FBM.*

$FBM_1 = m_5$

#S RG = 32

#S DZ = 17

#S LZ = 15

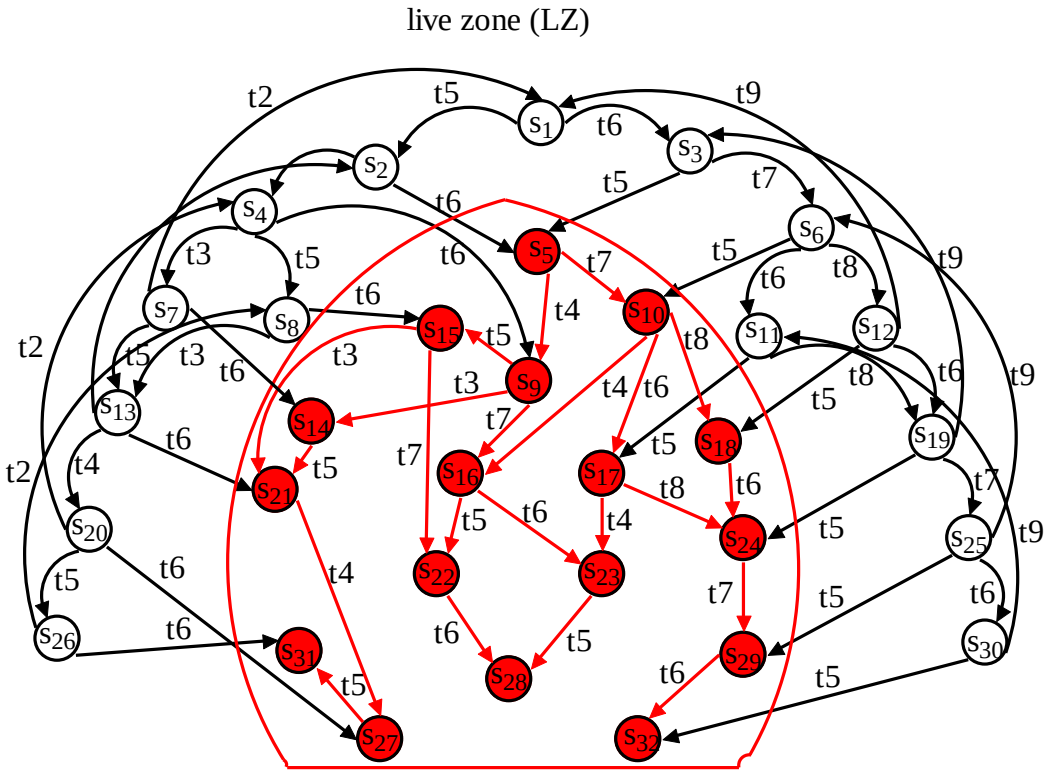

Figure 3.4. The Reachability graph of the *RPNM1*.

Table 3.1. FBM_1 obtained for *RPNM1*.

NODE	***p*3**	***p*4**	***p*5**	***p*7**	***p*8**	***p*9**	*p*11	*p*12	*p*13	*p*14
m_5	**0**	**0**	**1**	**1**	**0**	**0**	0	1	1	0

It can be seen that places *p5* and *p7* are the only marked activity places within m_5. This means that when there is a token each in *p5* and in *p7*, the net will be in DZ_1. Therefore, it is necessary to declare the total number of tokens in activity places *p5* and *p7* to be no more than 1 in order not to reach m_5. This constraint can be represented as a $PI = \mu_5 + \mu_7 \leq 1$. In order to compute the monitor ''C_1'', Eqs. (3.1) and (3.2) are used. It is obvious from Eq. (3.2) that the initial marking of the monitor $\mu_{c1(0)} = 1$ and that the PI_1 - related places are *p5* and *p7***.** This means that the PI_1 related net consists of *p5* and *p7* together with their input-output arcs as shown in Fig. 3.3.

$$PI_1 = \mu_5 + \mu_7 \leq 1$$

$$D_{C\,1} = -L_{PI1} \,.\, D_{PI1} \tag{3.1}$$

$$\mu_{c1(0)} = 1 \tag{3.2}$$

where L_{PI1} is related marked activity places. From the above constraint, L_{PI1} is given by:

$$L_{PI1} = \begin{matrix} p5 & p7 \end{matrix} \begin{bmatrix} 1 & 1 \end{bmatrix}$$

and D_{PI1} is the place invariant related incidence matrix. Therefore D_{PI1} is as follows:

$$D_{PI1} = \begin{matrix} \\ p5 \\ p7 \end{matrix} \begin{matrix} \begin{matrix} t4 & t5 & t6 & t7 \end{matrix} \\ \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \end{matrix} \tag{3.3}$$

Therefore $D_{C1} = -L_{PI1} \,.\, D_{PI1}$

$$D_{C1} = -\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

$$D_{C1} = -\begin{matrix} \begin{matrix} t4 & t5 & t6 & t7 \end{matrix} \\ \begin{bmatrix} -1 & 1 & 1 & -1 \end{bmatrix} \end{matrix}$$

$$D_{C1} = \begin{matrix} \begin{matrix} t4 & t5 & t6 & t7 \end{matrix} \\ \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix} \end{matrix}$$

The monitor *C1* is computed as shown in Table 3.2.

Table 3.2. Monitor (control place) *C1* computed for *RPNM1*.

FBM_1	PI_1		$^{\bullet}C_1$	$C_1^{\bullet}$	$\mu_0(C_1)$
$\mu_5 = 1, \mu_7 = 1$	$\mu_5 + \mu_7 \leq 1$	*C1*	*t*4, *t*7	*t*5, *t*6	1

The procedures are repeated for 5 iterations, and 5 necessary monitors are computed as shown in Table 3.3.

Table 3.3. Monitors computed for *RPNM1*.

FBM_i	PI_i	C_i	$^{\bullet}C_i$	$C_i^{\bullet}$	$\mu_0(C_i)$
$\mu_5 = 1, \mu_7 = 1$	$\mu_5 + \mu_7 \leq 1$	*C1*	*t*4, *t*7	*t*5, *t*6	1
$\mu_4 = 1, \mu_7 = 1$	$\mu_4 + \mu_7 \leq 1$	*C2*	*t*3, *t*7	*t*4, *t*6	1
$\mu_5 = 1, \mu_8 = 1$	$\mu_5 + \mu_8 \leq 1$	*C3*	*t*4, *t*8	*t*5, *t*7	1
$\mu_3 = 1, \mu_7 = 1$	$\mu_3 + \mu_7 \leq 1$	*C4*	*t*2, *t*7	*t*3, *t*6	1
$\mu_5 = 1, \mu_9 = 1$	$\mu_5 + \mu_9 \leq 1$	*C5*	*t*4, *t*9	*t*5, *t*8	1

Step 3.1.3 Place invariants are generalized as shown in Table 3.4 when $M_0(p_{11}) = M_0(p_{12}) = M_0(p_{13}) = M_0(p_{14}) = 1$.

Table 3.4. Generalized place invariants when one token is deposited in each shared resource place (p_{11}, p_{12}, p_{13} and p_{14}).

$$\mu_5 + \mu_7 \leq (\mu_{11} + \mu_{14}) - 1$$
$$\mu_4 + \mu_7 \leq (\mu_{11} + \mu_{13}) - 1$$
$$\mu_5 + \mu_8 \leq (\mu_{12} + \mu_{14}) - 1$$
$$\mu_3 + \mu_7 \leq (\mu_{11} + \mu_{12}) - 1$$
$$\mu_5 + \mu_9 \leq (\mu_{13} + \mu_{14}) - 1$$

For one token in each shared resource, after five iterations the procedure terminates. 5 monitors are computed as shown in Table 3.3. It can also be verified that the controlled *RPNM1*, obtained by adding these 5 control places to the uncontrolled *RPNM1*, is live with 15 good states. This is optimal live behavior for the controlled *RPNM*1. When we add five control places shown in Table 3.3 to the original *PNM1* shown in Fig. 3.5, we obtain the optimally controlled *PNM1* which is live and can reach all 31 good states.

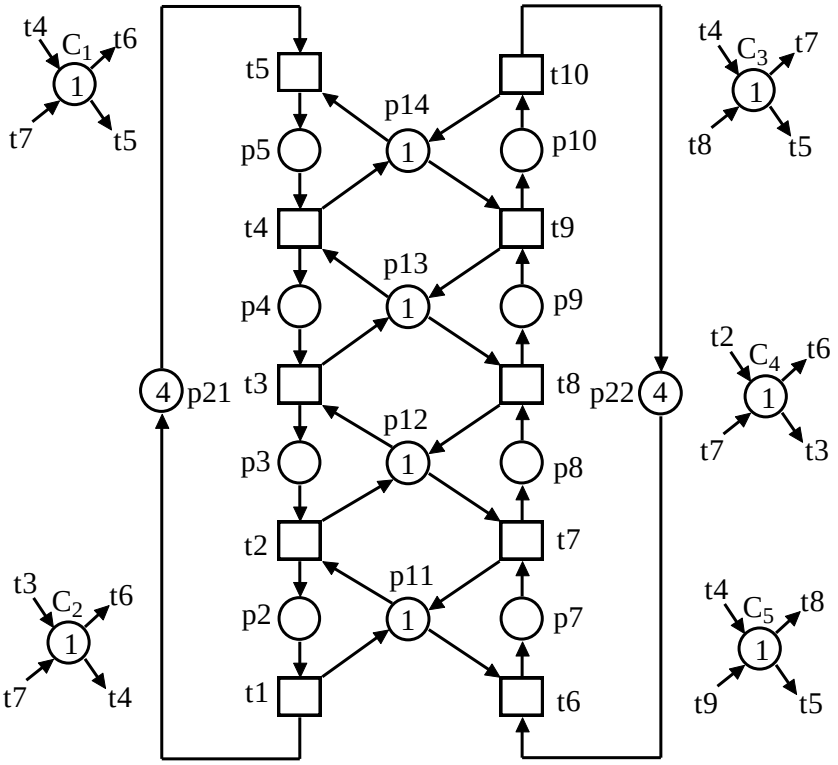

Figure 3.5. Optimally controlled *PNM1*.

Step 3.2.1 The number of tokens in the adjustable shared resource places are set to 2 ($M_0(p_{11}) = M_0(p_{12}) = M_0(p_{13}) = M_0(p_{14}) = 2$) as shown in Fig. 3.6.

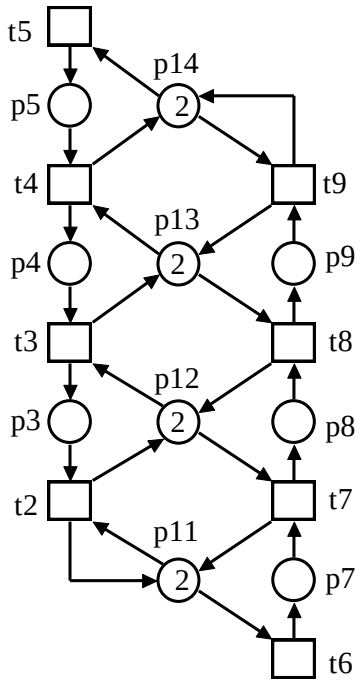

Figure 3.6. Reduced *PNM2* (*RPNM2*) with two tokens deposited in each shared resource place.

It is verified that the *RPNM2* shown in Fig 3.6 suffers from deadlocks.

Step 3.2.2 Algorithm 1 is used to compute monitors and the redundancy test is used to find necessary monitors.

It is seen that there are 315 states (*#S* in *RG*) in the RG_1 of the *RPNM*2. For the liveness property, the following is obtained:

```
The liveness: The net is not live.
The following nodes (i.e states) are not full: 116, 134, 145, 155,
175, 176, 186, 195, 196, 216, 217, 218, 225, 226, 234, 238, 246,
249, 254, 255, 257, 261, 267, 268, 270, 275, 280, 284, 285,
287,288,292,293, 294, 298, 300, 301, 304, 305, 307, 308, 309, 311,
313, 314, 315.
```

These 46 bad states (*#S* in *DZ*) constitute the DZ_1 and the remaining 269 good states (*#S* in *LZ*) represent the LZ_1. Therefore, an optimal deadlock-prevention policy should provide an answer to this problem such that 46 bad states are removed from the RG_1 and the remaining 269 good states are always reachable. Therefore the objective is to eliminate the DZ_1. In the first iteration from the DZ_1 (i = 0), the first marking namely $FBM_1 = m_{116}$ provided in Table 3.5 is chosen as an *FBM.*

$FBM_1 = m_{116}$

#S RG = 315

#S DZ = 46

#S LZ = 269

Table 3.5. FBM_1 obtained for *RPNM2*.

NODE	***p*3**	***p*4**	***p*5**	***p*7**	***p*8**	***p*9**	*p*11	*p*12	*p*13	*p*14
m_{116}	**2**	**0**	**0**	**2**	**0**	**0**	0	0	2	2

It can be seen that places *p3* and *p7* are the only marked activity places within m_{116}. This means that when there are 2 tokens both in *p3* and in *p7*, the net will be in DZ_1. Therefore, it is necessary to declare the total number of tokens in activity place *p3* and *p7* to be no more than 3 in order not to reach m_{116}. This constraint can be represented as a *PI* = $\mu_3 + \mu_7 \leq 3$. In order to compute the monitor ''C_1'', Eqs. (3.4) and (3.5) are used. It is obvious from Eq. (3.5) that the initial marking of the monitor $\mu_{c1(0)} = 3$ and that the PI_1 - related places are *p3* and *p7***.** This means that the PI_1 related net consists of *p3* and *p7* together with their input-output arcs as shown in Fig.3.6.

$$PI_1 = \mu_3 + \mu_7 \leq 3$$

$$D_{C\,1} = -L_{PI1}\,.\,D_{PI1} \tag{3.4}$$

$$\mu_{c1(0)} = 3 \tag{3.5}$$

where L_{PI1} is related marked activity places. From the above constraint, L_{PI} is given by:

$$L_{PI1} = \begin{matrix} p3 & p7 \end{matrix} \begin{bmatrix} 1 & 1 \end{bmatrix}$$

and D_{PI1} is the place invariant related incidence matrix. Therefore D_{PI1} is as follows:

$$D_{PI1} = \begin{matrix} \\ p3 \\ p7 \end{matrix} \begin{matrix} \begin{matrix} t2 & t3 & t6 & t7 \end{matrix} \\ \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \end{matrix} \tag{3.6}$$

Therefore $D_{C1} = -L_{PI1} \,.\, D_{PI1}$

$$D_{C1} = -[1 \quad 1] \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

$$D_{C1} = -\begin{matrix} \begin{matrix} t2 & t3 & t6 & t7 \end{matrix} \\ [-1 \quad 1 \quad 1 \quad -1] \end{matrix}$$

$$D_{C1} = \begin{matrix} \begin{matrix} t2 & t3 & t6 & t7 \end{matrix} \\ [1 \quad -1 \quad -1 \quad 1] \end{matrix}$$

The monitor *C1* is computed as shown in Table 3.6.

Table 3.6. Monitor (control place) *C1* computed for *RPNM2*.

FBM_1	PI_1		$^\bullet C_1$	$C_1^\bullet$	$\mu_0(C_1)$
$\mu_3 = 2, \mu_7 = 2$	$\mu_3 + \mu_7 \leq 3$	*C1*	*t*2, *t*7	*t*3, *t*6	3

The procedures are repeated for 6 iterations, and 6 necessary monitors are computed as shown in Table 3.7.

Table 3.7. Monitors for *RPNM2*.

FBM_i	PI_i	C_i	$^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
$\mu_3 = 2, \mu_7 = 2$	$\mu_3 + \mu_7 \leq 3$	*C1*	*t*2, *t*7	*t*3, *t*6	3
$\mu_4 = 2, \mu_8 = 2$	$\mu_4 + \mu_8 \leq 3$	*C2*	*t*3, *t*8	*t*4, *t*7	3
$\mu_5 = 2, \mu_9 = 2$	$\mu_5 + \mu_9 \leq 3$	*C3*	*t*4, *t*9	*t*5, *t*8	3
$\mu_3 = 1, \mu_4 = 2$ $\mu_7 = 2, \mu_8 = 1$	$\mu_3 + \mu_4 + \mu_7 + \mu_8 \leq 5$	*C4*	*t*2, *t*8	*t*4, *t*6	5
$\mu_4 = 1, \mu_5 = 2$ $\mu_8 = 2, \mu_9 = 1$	$\mu_4 + \mu_5 + \mu_8 + \mu_9 \leq 5$	*C5*	*t*3, *t*9	*t*5, *t*7	5
$\mu_3 = 1, \mu_4 = 1$ $\mu_5 = 2, \mu_7 = 2$ $\mu_8 = 1, \mu_9 = 1$	$\mu_3 + \mu_4 + \mu_5 + \mu_7 + \mu_8 + \mu_9 \leq 7$	*C6*	*t*2, *t*9	*t*5, *t*6	7

Step 3.2.3 Place invariants are generalized as shown in Table 3.8 when $M_0(p_{11}) = M_0(p_{12}) = M_0(p_{13}) = M_0(p_{14}) = 2$.

Table 3.8. Generalized place invariants when two tokens are deposited in each shared Resource place (p_{11}, p_{12}, p_{13} and p_{14}).

$\mu_3 + \mu_7 \leq (\mu_{11} + \mu_{12}) - 1$
$\mu_4 + \mu_8 \leq (\mu_{12} + \mu_{13}) - 1$
$\mu_5 + \mu_9 \leq (\mu_{13+} \mu_{14}) - 1$
$\mu_3 + \mu_4 + \mu_7 + \mu_8 \leq (\mu_{11} + \mu_{12} + \mu_{13}) - 1$
$\mu_4 + \mu_5 + \mu_8 + \mu_9 \leq (\mu_{12} + \mu_{13} + \mu_{14}) - 1$
$\mu_3 + \mu_4 + \mu_5 + \mu_7 + \mu_8 + \mu_9 \leq (\mu_{11} + \mu_{12} + \mu_{13} + \mu_{14}) - 1$

When two tokens are deposited in each shared resource, after 6 iterations the procedure terminates. Six monitors are computed as shown in Table 3.7. It can also be verified that the controlled *RPNM2*, obtained by adding these six control places to the uncontrolled *RPNM2*, is live with 269 good states. This is the optimal live behavior for the controlled *RPNM2*. When 6 monitors shown in Table 3.7 are added to the original *PNM2*, the optimally controlled *PNM2* is obtained, shown in Fig. 3.7 which is live and can reach all 1084 good states.

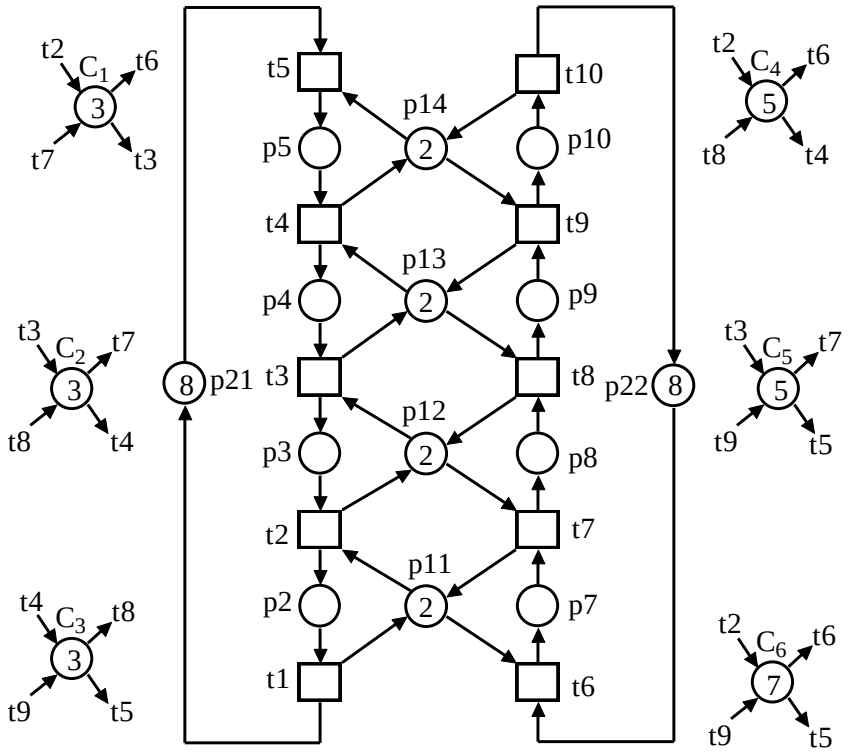

Figure 3.7. Optimally controlled *PNM2*.

Step 3.3.1 The number of tokens in the adjustable shared resource places are set to 3 ($M_0(p_{11}) = M_0(p_{12}) = M_0(p_{13}) = M_0(p_{14}) = 3$) as shown in Fig. 3.8.

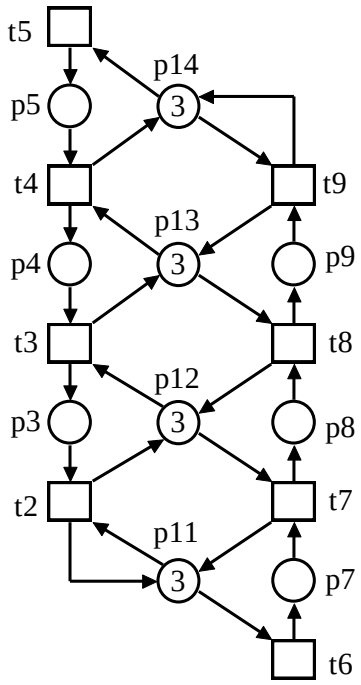

Figure 3.8. Reduced *PNM3* (*RPNM3*) with three tokens deposited in each shared resource place.

It is verified that the *RPNM3* shown in Fig 3.8 suffers from deadlocks.

Step 3.3.2 Algorithm 1 is used to compute monitors and the redundancy test is used to find necessary monitors.

It is seen that there are 1584 states (*#S* in *RG*) in the RG_1 of the *RPNM*3. For the liveness property, the following is obtained:

```
The liveness: The net is not live.
 The following nodes (i.e states) are not full: 568, 650, 690, 712,
801, 803, 839, 859, 861, 955, 957, 958, 989, 990, 1009,1010, 1040,
1091, 1104, 1106, 1107, 1108, 1132, 1134, 1150, 1152, 1181, 1204,
1229, 1241, 1242, 1243, 1246, 1262, 1263, 1277, 1278, 1284, 1297,
1303, 1329, 1348, 1349, 1358, 1359, 1361, 1366, 1372, 1373, 1383,
1385, 1389, 1399, 1403, 1403, 1416, 1420, 1429, 1442, 1443, 1450,
1451, 1454, 1458, 1459, 1466, 1467, 1469, 1475, 1478, 1489, 1494,
1501, 1509, 1510, 1515, 1516, 1518, 1519, 1523, 1524, 1525, 1529,
```

1536, 1539, 1542, 1543, 1548, 1551, 1552, 1555, 1556, 1558, 1559, 1560, 1562, 1566, 1568, 1569, 1572, 1573, 1575, 1577, 1578, 1580, 1582, 1583, 1584.

These 108 bad states (*#S* in *DZ*) constitute the DZ_1 and the remaining 1476 good states (*#S* in *LZ*) represent the LZ_1. Therefore, an optimal deadlock-prevention policy should provide an answer to this problem such that 108 bad states are removed from the RG_1 and the remaining 1476 good states are always reachable. Therefore our objective is to eliminate the DZ_1. In the first iteration from the DZ_1 (i = 0), the first marking namely $FBM_1 = m_{568}$ provided in Table 3.9 is chosen as an *FBM*.

$FBM_1 = m_{568}$

#S RG = 1584

#S DZ = 108

#S LZ = 1476

Table 3.9. FBM_1 obtained for *RPNM3*.

NODE	***p*3**	***p*4**	***p*5**	***p*7**	***p*8**	***p*9**	*p*11	*p*12	*p*13	*p*14
m_{568}	**3**	**0**	**0**	**3**	**0**	**0**	0	0	3	3

It can be seen that places *p3* and *p7* are the only marked activity places within m_{568}. This means that when there are 3 tokens both in *p3* and in *p7*, the net will be in DZ_1. Therefore, it is necessary to declare the total number of tokens in activity places *p3* and *p7* to be no more than 5 in order not to reach m_{568}. This constraint can be represented as a *PI* = $\mu_3 + \mu_7 \leq 5$. In order to compute the monitor ‘‘C_1’’, Eqs. (3.7) and (3.8) are used. It is obvious from Eq. (3.8) that the initial marking of the monitor $\mu_{c1(0)} = 5$ and that the PI_1-related places are *p3* and *p7***.** This means that the PI_1 related net consists of *p3* and *p7* together with their input-output arcs as shown in Fig. 3.8.

$$PI_1 = \quad \mu_3 + \mu_7 \leq 5$$

$$D_{C\,1} = -L_{PI1} \,.\, D_{PI1} \tag{3.7}$$

$$\mu_{c1(0)} = 5 \tag{3.8}$$

where L_{PI1} is related marked activity places. From the above constraint, L_{PI} is given by:

$$L_{PI} = \begin{matrix} p3 & p7 \end{matrix} \begin{bmatrix} 1 & 1 \end{bmatrix}$$

and D_{PI1} is the place invariant related incidence matrix. Therefore D_{PI1} is as follows:

$$D_{PI1} = \begin{matrix} \\ p3 \\ p7 \end{matrix} \begin{matrix} t2 & t3 & t6 & t7 \end{matrix} \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \quad (3.9)$$

Therefore $D_{C1} = -L_{PI1} \,.\, D_{PI1}$

$$D_{C1} = -\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix}$$

$$D_{C1} = -\begin{matrix} t2 & t3 & t6 & t7 \end{matrix} \begin{bmatrix} -1 & 1 & 1 & -1 \end{bmatrix}$$

$$D_{C1} = \begin{matrix} t2 & t3 & t6 & t7 \end{matrix} \begin{bmatrix} 1 & -1 & -1 & 1 \end{bmatrix}$$

The monitor *C1* is computed as shown in Table 3.10.

Table 3.10. Monitor (control place) *C1* computed for *RPNM3*.

FBM_1	PI_1		$^{\bullet}C_1$	$C_1^{\bullet}$	$\mu_0(C_1)$
$\mu_3 = 3, \mu_7 = 3$	$\mu_3 + \mu_7 \leq 5$	*C1*	*t2*, *t7*	*t3*, *t6*	5

The procedures are repeated for 6 iterations, and 6 necessary monitors are computed as shown in Table 3.11.

Table 3.11. Monitors computed for *RPNM3*.

BM_i	PI_i	C_i	$^{\bullet}C_i$	$C_i^{\bullet}$	$\mu_0(C_i)$
$\mu_3 = 3, \mu_7 = 3$	$\mu_3 + \mu_7 \leq 5$	*C1*	*t*2, *t*7	*t*3, *t*6	5
$\mu_4 = 3, \mu_8 = 3$	$\mu_4 + \mu_8 \leq 5$	*C2*	*t*3, *t*8	*t*4, *t*7	5
$\mu_5 = 3, \mu_9 = 3$	$\mu_5 + \mu_9 \leq 5$	*C3*	*t*4, *t*9	*t*5, *t*8	5
$\mu_3 = 2, \mu_4 = 3$ $\mu_7 = 3, \mu_8 = 1$	$\mu_3 + \mu_4 + \mu_7 + \mu_8 \leq 8$	*C4*	*t*2, *t*8	*t*4, *t*6	8
$\mu_4 = 1, \mu_5 = 3$ $\mu_8 = 3, \mu_9 = 2$	$\mu_4 + \mu_5 + \mu_8 + \mu_9 \leq 8$	*C5*	*t*3, *t*9	*t*5, *t*7	8
$\mu_3 = 2, \mu_4 = 2$ $\mu_5 = 3, \mu_7 = 3$ $\mu_8 = 1, \mu_9 = 1$	$\mu_3 + \mu_4 + \mu_5 + \mu_7 + \mu_8 + \mu_9 \leq 11$	*C6*	*t*2, *t*9	*t*5, *t*6	11

Step 3.3.3 Place invariants are generalized as shown in Table 3.12, when $M_0(p_{11}) = M_0(p_{12}) = M_0(p_{13}) = M_0(p_{14}) = 3$

Table 3.12. Generalized place invariants when three token is deposited in each shared resource place (p_{11}, p_{12}, p_{13} and p_{14}).

$$\mu_3 + \mu_7 \leq (\mu_{11} + \mu_{12}) - 1$$
$$\mu_4 + \mu_8 \leq (\mu_{12} + \mu_{13}) - 1$$
$$\mu_5 + \mu_9 \leq (\mu_{13+} \mu_{14}) - 1$$
$$\mu_3 + \mu_4 + \mu_7 + \mu_8 \leq (\mu_{11} + \mu_{12} + \mu_{13}) - 1$$
$$\mu_4 + \mu_5 + \mu_8 + \mu_9 \leq (\mu_{12} + \mu_{13} + \mu_{14}) - 1$$
$$\mu_3 + \mu_4 + \mu_5 + \mu_7 + \mu_8 + \mu_9 \leq (\mu_{11} + \mu_{12} + \mu_{13} + \mu_{14}) - 1$$

The last computation is carried out for the case in which each shared resource holds 3 tokens. After 6 iterations the procedure terminates. Six necessary monitors are computed as shown in Table 3.11. It can also be verified that the controlled *RPNM3*, obtained by adding these 6 monitors to the uncontrolled *RPNM3*, is live with 1476 good state. This is the optimal live behavior for the controlled *RPNM3*. When 6 monitors shown in Table 3.11 are added to the original Petri net model, the optimally controlled *PNM3* is obtained shown in Fig. 3.9 which is live and can reach all 9362 good states.

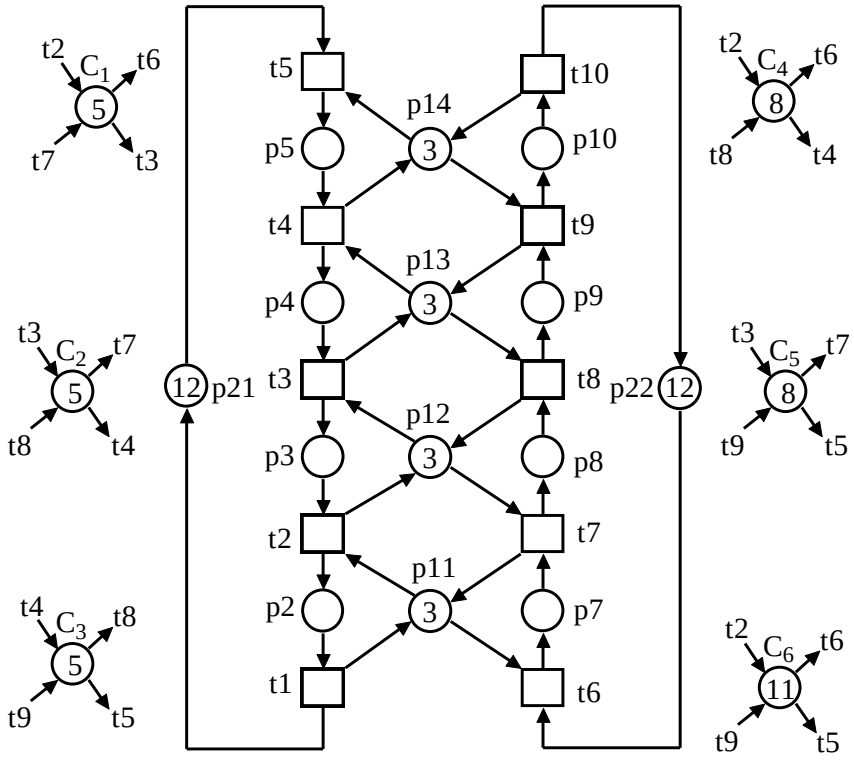

Figure 3.9. Optimally controlled *PNM3*.

3.1.1 DISCUSSION

When there is only one token each in the shared resource places *p11*, *p12*, *p13* and *p14* the generalized *PIs* shown in Table 3.4 are valid, On the other hand, it is obvious that when there are two or three tokens in each shared resource place (*p11*, *p12*, *p13* and *p14*) generalized *PIs* are the same as depicted in Tables 3.8 and 3.12. This means that these generalized *PIs* are valid for the instances $p11 = p12 = p13 = p14 = N$, $N = 2, 3, 4, \ldots$.

3.1.2 REACHABILITY STATES COMPARISON BETWEEN THE ORIGINAL PNM AND THE REDUCED PNM

Tables 3.13 and 3.14 present numerical experiments to compare the live states generated in *RPNM* and the original *PNM* for different set of values of tokens in the shared resource places (p_{11}, p_{12}, p_{13} and p_{14}). These results verify the correctness of generalized *PIs* obtained.

Table 3.13. Analysis results for the *RPNM* with different instances of number of tokens in the shared resource places.

Instance	Reduced net			Controlled net
(μ_{11}, μ_{12}, μ_{13}, μ_{14})	N_{RG}	N_{DZ}	N_{LZ}	N_{RG}
(1,1,1,1)	32	17	15	15
(2,2,2,2)	315	46	269	269
(3,3,3,3)	1584	108	1476	1476
(4,4,4,4)	5600	204	5396	5396
(5,5,5,5)	15840	340	15500	15500
(6,6,6,6)	3836	522	37845	37845
(7,7,7,7)	82880	756	82124	82124
(8,8,8,8)	163944	1048	162896	162896
(9,9,9,9)	302400	1404	300996	300996
(10,10,10,10)	526955	1830	525125	525125
(11,11,11,11)	875952	2332	873620	873620
(12,12,12,12)	1399320	2916	1396404	1396404
(13,13,13,13)	2160704	3588	2157116	2157116

Table 3.14. Analysis results for the *PNM* with different instances of number of tokens in the shared resource places.

Instance	Original net			Controlled net
(μ_{11}, μ_{12}, μ_{13}, μ_{14})	N_{RG}	N_{DZ}	N_{LZ}	N_{RG}
(1,1,1,1)	48	17	31	31
(2,2,2,2)	1176	92	1084	1084
(3,3,3,3)	9657	295	9362	9362
(4,4,4,4)	49852	701	49151	49151
(5,5,5,5)	192975	406	191569	191569
(6,6,6,6)	612000	2524	609476	609476
(7,7,7,7)	1675261	4187	1671074	1671074
(8,8,8,8)	-		-	-
(9,9,9,9)	-		-	-
(10,10,10,10)	-		-	-
(11,11,11,11)	-		-	-
(12,12,12,12)	-		-	-
(13,13,13,13)	-		-	-

NOTE : The dash line indicates the number of states within the reachability graph (N_{RG}) and live zone (N_{LZ}) are too large to be computed, i.e., they have than 2 million states.

CHAPTER 4

APPLICATION EXAMPLES

4.1 PETRI NET MODEL ANALYSIS

In this section the Petri net model (*PNM*) shown in Fig. 4.1 is considered for different scenarios. This *PNM* suffers from deadlock problems. In the reachability graph (*RG*) of this PNM there are 282 states, 77 of which are in the *DZ* while 205 of which are in the *LZ*. The Petri net model of Fig. 4.1 have one token in each shared resource place (*p21*, *p22* and *p23*). So one token in each shared resource is common in all scenarios. The scenarios are categorized as follows:

Scenario 1: The number of tokens in the adjustable shared resource places are set to the same values: N, $N = 1, 2, 3$.

Scenario 2: The number of tokens in the adjustable shared resource places are set as follows: $p21 = p22 = 1$ are always constant while $p23 = N$, $N = 1, 2, 3$.

Scenario 3: The number of tokens in the adjustable shared resource places are set as follows: $p22 = p23 = 1$ are always constant while $p21 = N$, $N = 1, 2, 3$.

Scenario 4: The number of tokens in the adjustable shared resource places are set as follows: $p21 = p23 = 1$ are always constant while $p22 = N$, $N = 1, 2, 3$.

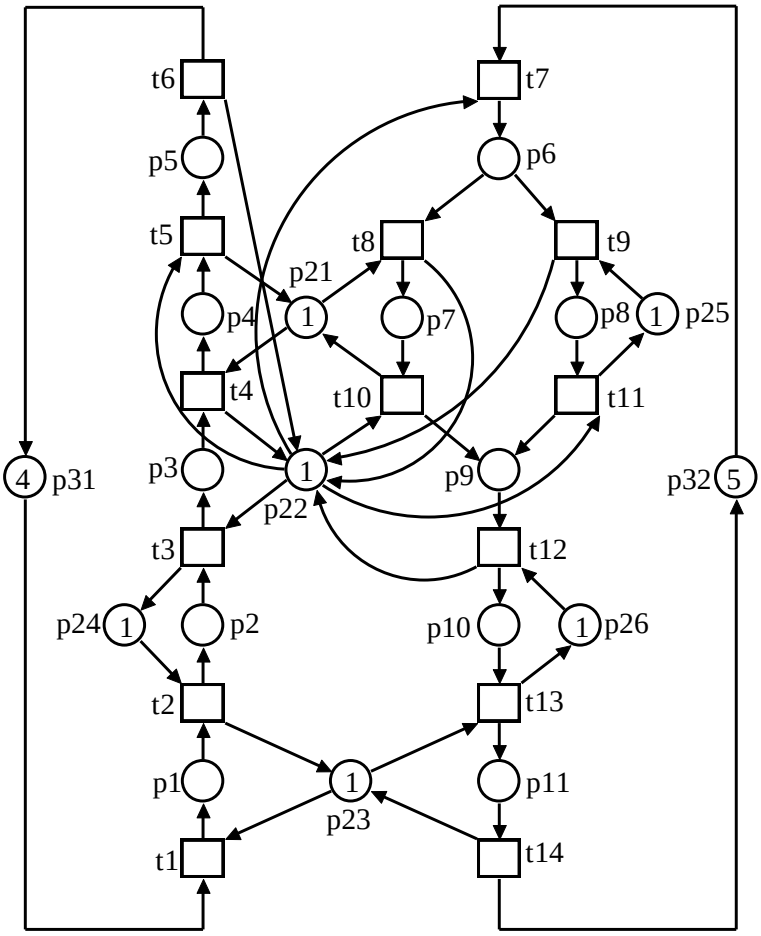

Figure 4.1. Petri net model of the system (S^4PR net).

4.1.1 TRANSFORMATION OF ORIGINAL PNM INTO REDUCED PNM

In order to obtain the reduced *PNM*, sink/source places, namely *p31* and *p32* of Fig. 4.1 are removed together with their input/output arcs. It can be verified that the removal of the places does not affect the behavior of the *PNM*. Both production sequences can be carried out as described before. When *p31* and *p32* are removed together with their input/output arcs, the *PNM* provides the same behavior. In this case, properties such as liveness, safeness and reversibility are preserved. Secondly, by using Petri net reduction

rules, series transition *t5* and *t6* (*t13* and *t14* respectively) can be merged as *t5* (*t13* respectively). Finally, the reduced *PNM* (*RPNM*) is obtained as shown in Fig. 4.2.

4.1.1.2 The adjustable shared resource places are *p21*, *p22* and *p23*.

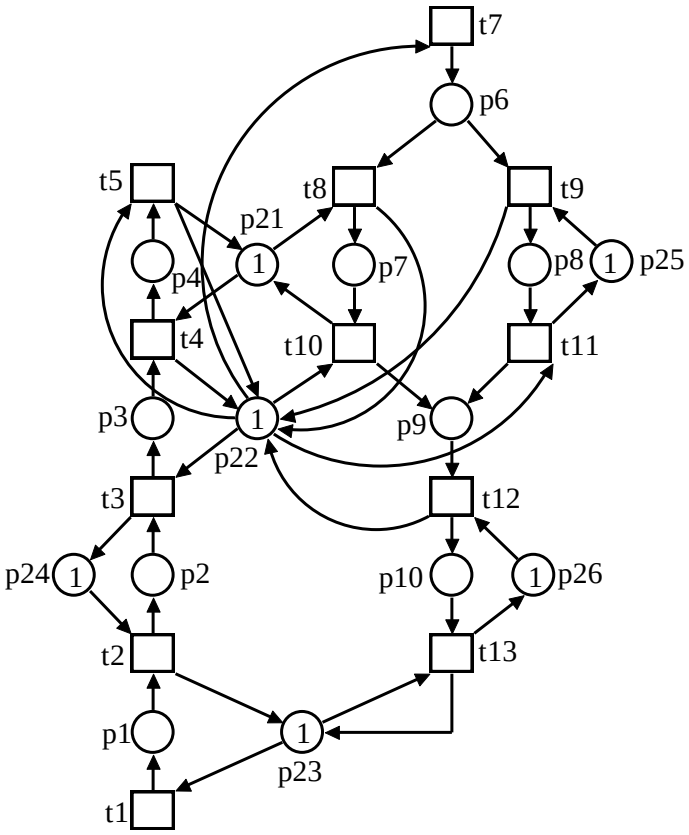

Figure 4.2. The Reduced *PNM* (*RPNM*).

4.1.2 SCENARIO 1 (COMPUTATION OF MONITORS)

The computation of monitors are carried out based on the scenario 1: The number of tokens in the adjustable shared resource places are set to the same values: $M_0(p_{21}) = M_0(p_{22}) = M_0(p_{23}) = N$, $N = 1, 2, 3$.

4.1.2.1 The number of tokens in the adjustable shared resource places are set to 1 ($M_0(p_{21})$ = $M_0(p_{22})$ = $M_0(p_{23})$ = 1) as shown in Fig. 4.3.

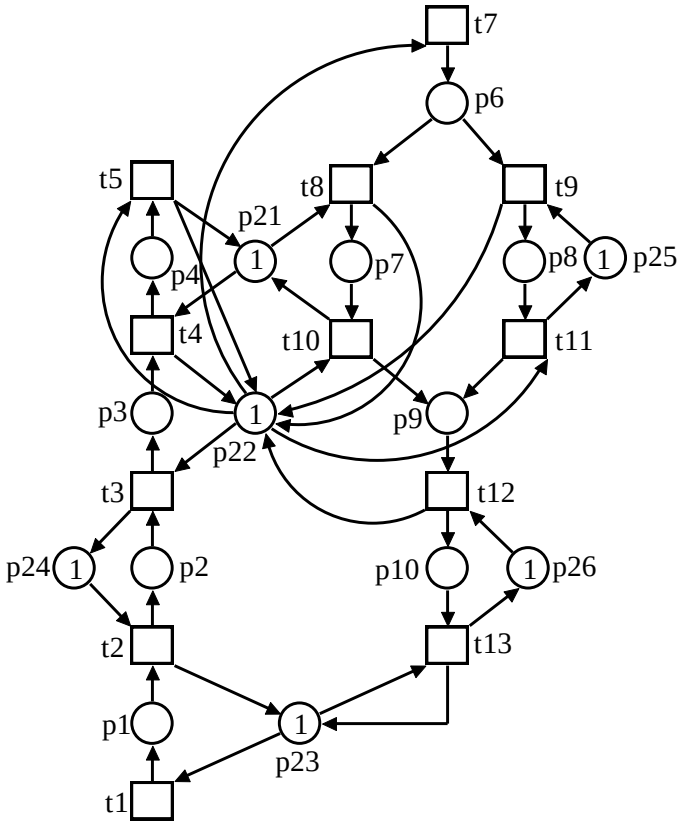

Figure 4.3. Reduced (*RPNM1*) with one token deposited in each shared resource place.

4.1.2.2 8 necessary monitors shown in Table 4.1 are computed after 8 iterations for the *RPNM* shown in Fig. 4.3.

Table 4.1. Monitors computed for the *RPNM1*.

FBM_i	PI_i	C_i	$^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
$\mu_3 = 1, \mu_7 = 1$	$\mu_3 + \mu_7 \leq 1$	*C1*	*t*4, *t*10	*t*3, *t*8	1
$\mu_6 = 1, \mu_7 = 1, \mu_8 = 1$	$\mu_6 + \mu_7 + \mu_8 \leq 2$	*C2*	*t*10, *t*11	*t*7	2
$\mu_3 = 1, \mu_4 = 1$	$\mu_3 + \mu_4 \leq 1$	*C3*	*t*5	*t*3	1
$\mu_4 = 1, \mu_6 = 1, \mu_8 = 1$	$\mu_4 + \mu_6 + \mu_8 \leq 2$	*C4*	*t*5, *t*8, *t*11	*t*4, *t*7	2
$\mu_1 = 1, \mu_2 = 1, \mu_7 = 1, \mu_9 = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq 3$	*C5*	*t*3, *t*12	*t*5, *t*8, *t*11	3
$\mu_1 = 1, \mu_2 = 1, \mu_7 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq 3$	*C6*	*t*3, *t*10, *t*13	*t*1, *t*8, *t*12	3
$\mu_1 = 1, \mu_2 = 1, \mu_6 = 1, \mu_8 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_6 + \mu_8 + \mu_{10} \leq 4$	*C7*	*t*3, *t*8, *t*11, *t*13	*t*1, *t*7, *t*12	4
$\mu_1 = 1, \mu_2 = 1, \mu_9 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq 3$	*C8*	*t*3, *t*13	*t*1, *t*10, *t*11	3

4.1.2.3 Place invariants are generalized as shown in Table 4.2 when $M_0(p_{21}) = M_0(p_{22}) = M_0(p_{23}) = 1$.

Table 4.2. Generalized place invariants when one token is deposited in each shared resource place (p_{21}, p_{22}, and p_{23}).

$$\mu_3 + \mu_7 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_6 + \mu_7 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_3 + \mu_4 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_4 + \mu_6 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq (\mu_{21} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_6 + \mu_8 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{25} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$

4.1.2.4 For one token in each shared resource place, after eight iterations the procedure terminates. 8 monitors are computed as shown in Table 4.1. It can also be verified that the controlled *RPNM1*, obtained by adding these 8 monitors to the uncontrolled *RPNM*, is live with 119 good states. This is optimal live behavior for the controlled *RPNM1*. When 8

monitors shown in Table 4.1 are added to the original *PNM1* the optimally controlled *PNM1*, shown in Fig. 4.4, which is live and can reach all 205 good states.

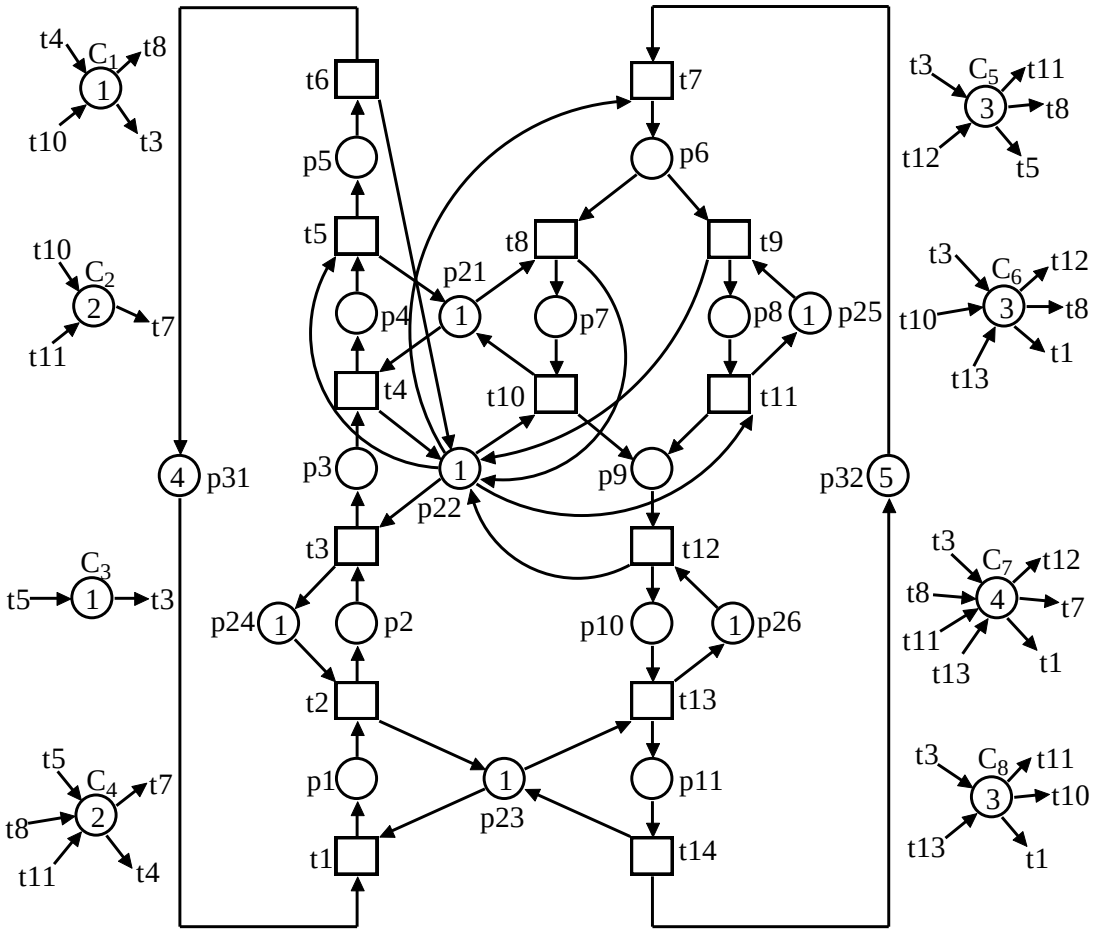

Figure 4.4. Optimally controlled *PNM1*.

4.1.3.1 The number of tokens in the adjustable shared resource places are all set to 2 ($M_0(p_{21}) = M_0(p_{22}) = M_0(p_{23}) = 2$) as shown in Fig. 4.5

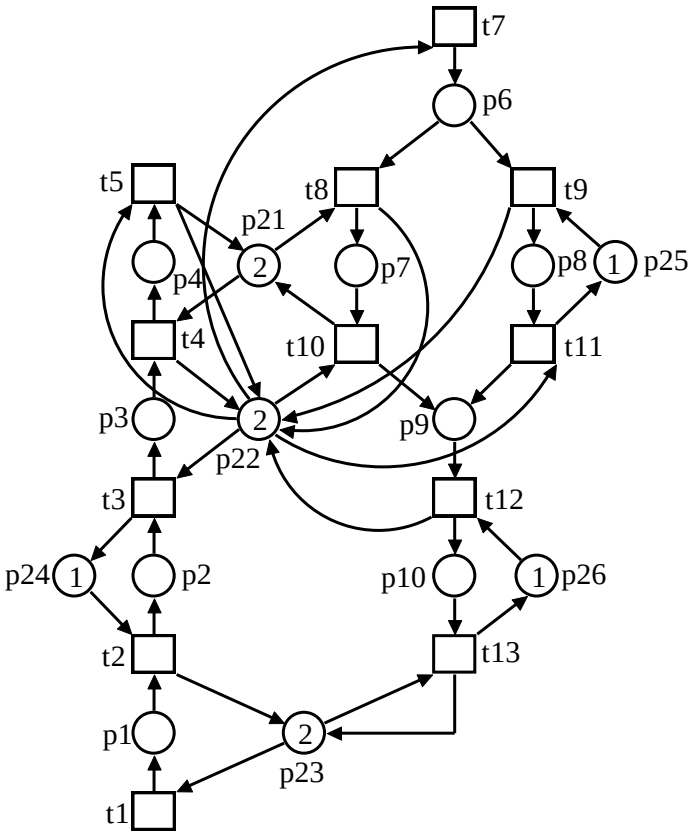

Figure 4.5. Reduced *PNM2* (*RPNM2*) with two tokens deposited in each shared resource place.

4.1.3.2 5 necessary monitors shown in Table 4.3 are computed after 11 iterations for the *RPNM2* shown in Fig. 4.5.

Table 4.3. Monitors computed for the *RPNM2*.

FBM_i	PI_i	C_i	$^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
$\mu_3 = 2$, $\mu_4 = 1$, $\mu_7 = 1$,	$\mu_3 + \mu_4 + \mu_7 \leq 3$	*C1*	*t*5, *t*10	*t*3, *t*8	3
$\mu_3 = 1$, $\mu_4 = 1$, $\mu_6 = 1$, $\mu_7 = 1$, $\mu_8 = 1$	$\mu_3 + \mu_4 + \mu_6 + \mu_7 + \mu_8 \leq 4$	*C2*	*t*5, *t*10, *t*11	*t*3, *t*7	4
$\mu_1 = 2$, $\mu_2 = 1$, $\mu_9 = 2$, $\mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq 5$	*C3*	*t*3, *t*13	*t*1, *t*10, *t*11	5
$\mu_1 = 2$, $\mu_2 = 1$, $\mu_3 = 1$, $\mu_4 = 1$, $\mu_7 = 1$, $\mu_9 = 1$, $\mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_7 + \mu_9 + \mu_{10} \leq 7$	*C4*	*t*5, *t*13	*t*1, *t*8, *t*11	7
$\mu_1 = 2$, $\mu_2 = 1$, $\mu_4 = 1$, $\mu_6 = 1$, $\mu_7 = 1$, $\mu_8 = 1$, $\mu_9 = 1$, $\mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_4 + \mu_6 + \mu_7 + \mu_8 + \mu_9 + \mu_{10} \leq 8$	*C5*	*t*3, *t*5, *t*13	*t*1, *t*4, *t*7	8

4.1.3.3 Place invariants are generalized as shown in Table 4.4 when $M_0(p_{21}) = M_0(p_{22}) = M_0(p_{23}) = 2$.

Table 4.4. Generalized place invariants when two tokens are deposited in each shared resource place (p_{21}, p_{22}, and p_{23}).

$$\mu_3 + \mu_4 + \mu_7 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_3 + \mu_4 + \mu_6 + \mu_7 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_7 + \mu_9 + \mu_{10} \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_4 + \mu_6 + \mu_7 + \mu_8 + \mu_9 + \mu_{10} \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24} + \mu_{25} + \mu_{26}) - 1$$

4.1.3.4 When two tokens are deposited in each shared resource place, after 11 iterations the procedure terminates. Five monitors are computed as shown in Table 4.3. It can also be verified that the controlled *RPNM2*, obtained by adding these five control places to the uncontrolled *RPNM2*, is live with 1242 good states. This is the optimal live behavior for the controlled *RPNM2*. When 5 monitors shown in Table 4.3 are added to the original *PNM2*, the optimally controlled *PNM2* is obtained, shown in Fig. 4.6 which is live and can reach all 3711 good states.

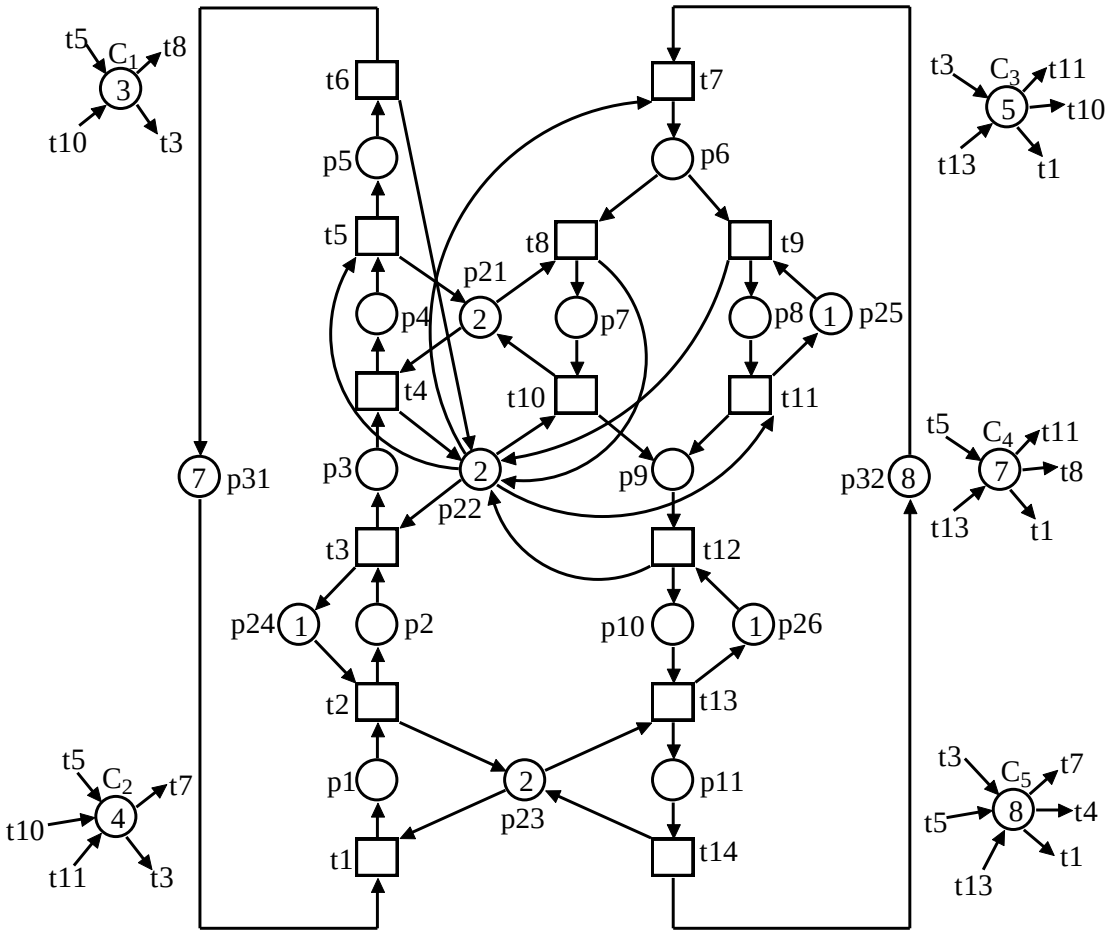

Figure 4.6. Optimally controlled *PNM2*.

4.1.4.1 The number of tokens in the adjustable shared resource places are all set to 3 ($M_0(p_{21}) = M_0(p_{22}) = M_0(p_{23}) = 3$) as shown in Fig. 4.7

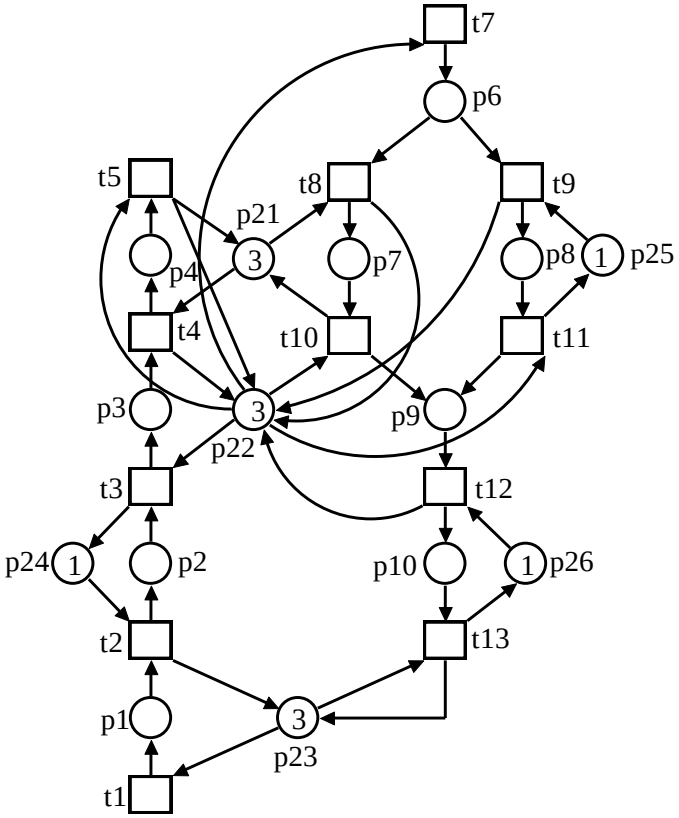

Figure 4.7. Reduced *PNM3* (*RPNM3*) with three tokens deposited in each shared resource place.

4.1.4.2 5 necessary monitors shown in Table 4.5 are computed after 13 iterations for the *RPNM3* shown in Fig. 4.7.

Table 4.5. Monitors computed for the *RPNM3*.

FBM_i	PI_i	C_i	$^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
$\mu_3 = 1, \mu_4 = 1, \mu_6 = 2, \mu_7 = 2, \mu_8 = 1$	$\mu_3 + \mu_4 + \mu_6 + \mu_7 + \mu_8 \leq 6$	*C1*	*t*5, *t*10, *t*11	*t*3, *t*7	6
$\mu_3 = 3, \mu_4 = 1, \mu_7 = 2,$	$\mu_3 + \mu_4 + \mu_7 \leq 5$	*C2*	*t*5, *t*10	*t*3, *t*8	5
$\mu_1 = 3, \mu_2 = 1, \mu_9 = 3, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq 7$	*C3*	*t*3, *t*13	*t*1, *t*10, *t*11	7
$\mu_1 = 3, \mu_2 = 1, \mu_3 = 2, \mu_4 = 1, \mu_7 = 2, \mu_9 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_7 + \mu_9 + \mu_{10} \leq 10$	*C4*	*t*5, *t*13	*t*1, *t*8, *t*11	10
$\mu_1 = 3, \mu_2 = 1, \mu_3 = 1, \mu_4 = 1, \mu_6 = 1, \mu_7 = 2, \mu_8 = 1, \mu_9 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_6 + \mu_7 + \mu_8 + \mu_9 + \mu_{10} \leq 11$	*C5*	*t*5, *t*13	*t*1, *t*7	11

4.1.4.3 Place invariants are generalized as shown in Table 4.6 when $M_0(p_{21}) = M_0(p_{22}) = M_0(p_{23}) = 3$.

Table 4.6. Generalized place invariants when three tokens are deposited in each shared resource place (p_{21}, p_{22}, and p_{23}).

$\mu_3 + \mu_4 + \mu_6 + \mu_7 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$
$\mu_3 + \mu_4 + \mu_7 \leq (\mu_{21} + \mu_{22}) - 1$
$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$
$\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_7 + \mu_9 + \mu_{10} \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$
$\mu_1 + \mu_1 + \mu_3 + \mu_4 + \mu_6 + \mu_7 + \mu_8 + \mu_9 + \mu_{10} \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24} + \mu_{25} + \mu_{26}) - 1$

4.1.4.4 The last computation is carried out for the case in which each shared resource place holds 3 tokens. After 13 iterations the procedure terminates. Five necessary monitors are computed as shown in Table 4.5. It can also be verified that the controlled *RPNM3*, obtained by adding these 5 monitors to the uncontrolled *RPNM3*, is live with 5972 good state. This is the optimal live behavior for the controlled *RPNM3*. When 5 monitors shown in Table 4.5 are added to the original *PNM3*, the optimally controlled *PNM3* is obtained shown in Fig. 4.8, which is live and can reach all 26316 good states.

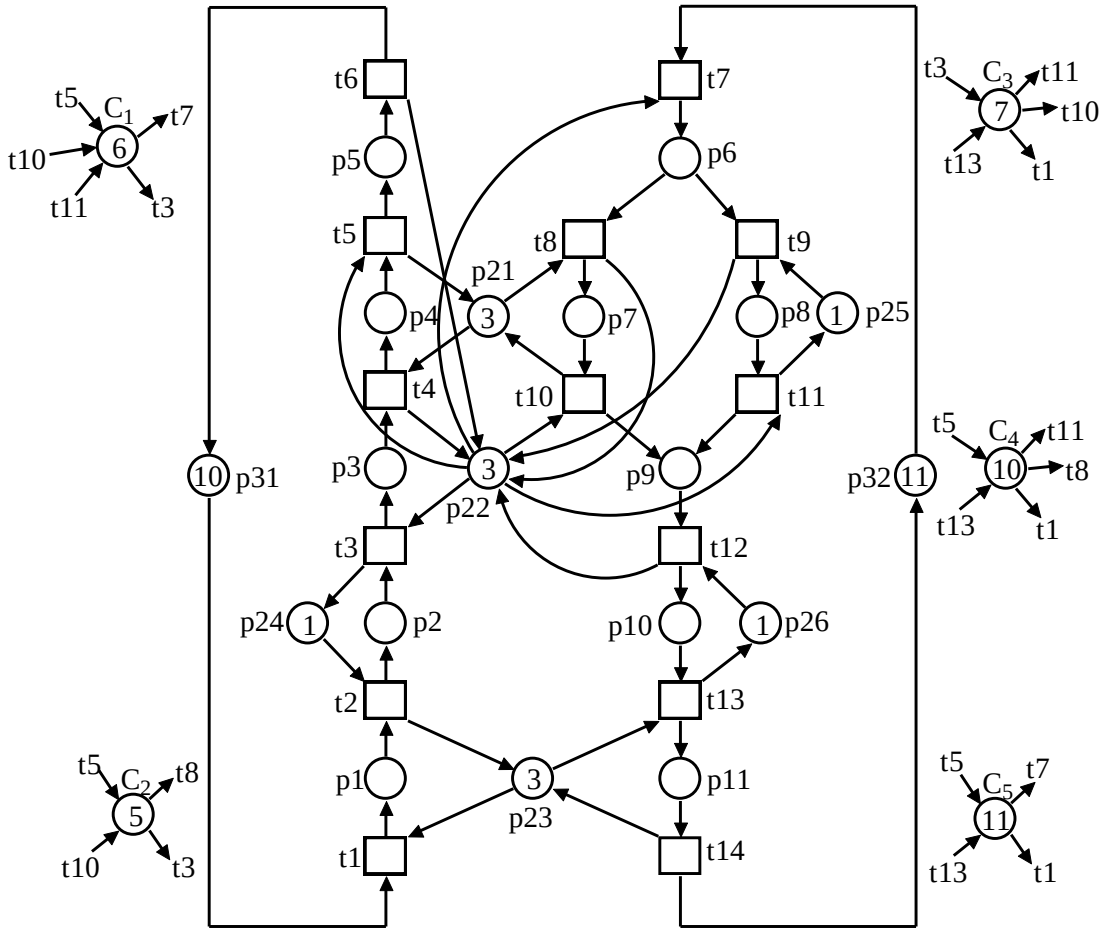

Figure 4.8. Optimally controlled *PNM3*.

4.1.5 DISCUSSION FOR THE SCENARIO 1

When there is only one token each in shared resource places *p21*, *p22* and *p23* the generalized *PIs* shown in Table 4.2 are valid, also when there are two tokens each in shared resource places *p21*, *p22* and *p23* the generalized *PIs* shown in Table 4.4 are valid. On the other hand, it is obvious that when there are three or four tokens in each shared resource place (*p21*, *p22* and *p23*) generalized *PIs* are the same as depicted in Table 4.6. This means that these generalized *PIs* are valid for the instances $M_0(p_{21}) = M_0(p_{22}) = M_0(p_{23}) = N$, $N = 3, 4, 5, \ldots$.

4.1.6 REACHABILITY STATES COMPARISON BETWEEN THE ORIGINAL PNM AND THE REDUCED PNM FOR THE SCENARIO 1

Tables 4.7 and 4.8 present numerical experiments to compare the live states generated in the *RPNM* and the original *PNM* for different set of values of tokens in the shared resource places (p_{21}, p_{22} and p_{23}). These results verify the correctness of generalized *PIs* obtained.

Table 4.7. Analysis results for the *RPNM* with different instances of the number of tokens in the shared resource places.

Instance (μ_{21}, μ_{22}, μ_{23})	Reduced net			Controlled net
	N_{RG}	N_{DZ}	N_{LZ}	N_{RG}
(1,1,1)	176	57	119	119
(2,2,2)	1406	164	1242	1242
(3,3,3)	6336	364	5972	5972
(4,4,4)	20900	685	20215	20215
(5,5,5)	56304	1152	55152	55152
(6,6,6)	131516	1792	129724	129724
(7,7,7)	276224	2632	273592	273592
(8,8,8)	534276	3699	530577	530577
(9,9,9)	967600	5020	962580	962580

Table 4.8. Analysis results for the *PNM* with different instances of number of tokens in the shared resource places.

Instance (μ_{21}, μ_{22}, μ_{23})	Original net			controlled net
	N_{RG}	N_{DZ}	N_{LZ}	N_{RG}
(1,1,1)	282	77	205	205
(2,2,2)	4011	300	3711	3711
(3,3,3)	27152	836	26316	26316
(4,4,4)	124110	1875	122235	122235
(5,5,5)	440850	3660	437190	437190
(6,6,6)	1310617	6482	1304135	1304135
(7,7,7)	-		-	-
(8,8,8)	-		-	-
(9,9,9)	-		-	-

NOTE : The dash line indicates the number of states within the reachability graph (N_{RG}) and live zone (N_{LZ}) are too large to be computed, i.e., they have more than 1 million states.

4.1.7 SCENARIO 2 (COMPUTATION OF MONITORS)

The computation of monitors are carried out based on the scenario 2: The number of tokens in the adjustable shared resource places are set as follows: $M_0(p_{21}) = M_0(p_{22}) = 1$ are always constant while $M_0(p_{23}) = N$, $N = 1, 2, 3$.

4.1.7.1 The number of tokens in the adjustable shared resource places are set to 1 ($M_0(p_{21}) = M_0(p_{22}) = M_0(p_{23}) = 1$) as shown in Fig. 4.9.

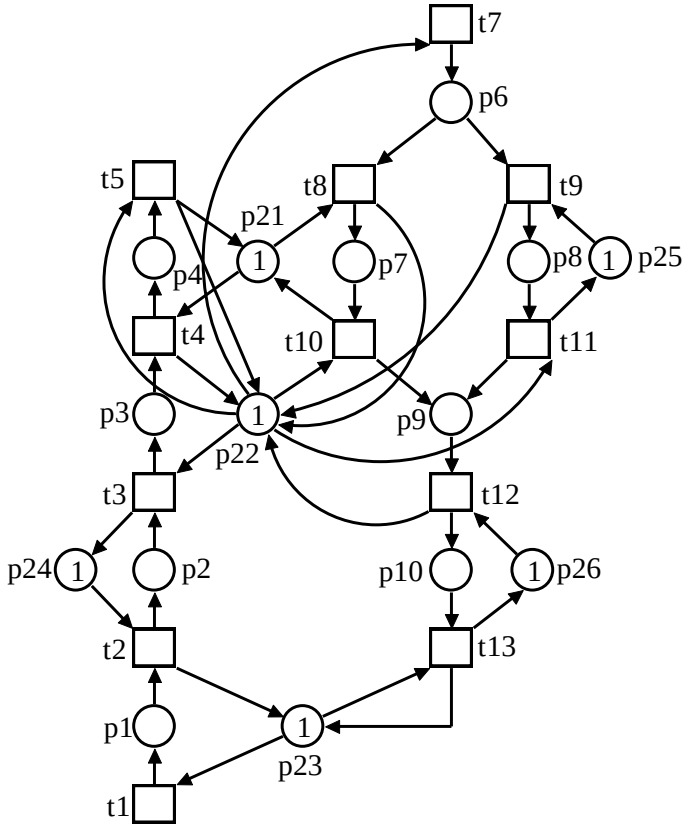

Figure 4.9. Reduced *PNM1* (*RPNM1*) with one token deposited in each shared resource place.

4.1.7.2 8 necessary monitors shown in Table 4.9 are computed after 8 iterations for the *RPNM* shown in Fig. 4.9.

Table 4.9. Monitors computed for the *RPNM1*.

FBM_i	PI_i	C_i	$^{\bullet}C_i$	$C_i^{\bullet}$	$\mu_0(C_i)$
$\mu_3 = 1$, $\mu_7 = 1$	$\mu_3 + \mu_7 \leq 1$	*C1*	*t*4, *t*10	*t*3, *t*8	1
$\mu_6 = 1$,$\mu_7 = 1$, $\mu_8 = 1$	$\mu_6 + \mu_7 + \mu_8 \leq 2$	*C2*	*t*10, *t*11	*t*7	2
$\mu_3 = 1$, $\mu_4 = 1$	$\mu_3 + \mu_4 \leq 1$	*C3*	*t*5	*t*3	1
$\mu_4 = 1$,$\mu_6 = 1$, $\mu_8 = 1$	$\mu_4 + \mu_6 + \mu_8 \leq 2$	*C4*	*t*5, *t*8, *t*11	*t*4, *t*7	2
$\mu_1 = 1$,$\mu_2 = 1$, $\mu_7 = 1$, $\mu_9 = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq 3$	*C5*	*t*3, *t*12	*t*5, *t*8, *t*11	3
$\mu_1 = 1$,$\mu_2 = 1$, $\mu_7 = 1$,$\mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq 3$	*C6*	*t*3, *t*10, *t*13	*t*1, *t*8, *t*12	3
$\mu_1 = 1$,$\mu_2 = 1$, $\mu_6 = 1$, $\mu_8 = 1$, $\mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_6 + \mu_8 + \mu_{10} \leq 4$	*C7*	*t*3, *t*8, *t*11, *t*13	*t*1, *t*7, *t*12	4
$\mu_1 = 1$,$\mu_2 = 1$, $\mu_9 = 1$,$\mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq 3$	*C8*	*t*3, *t*13	*t*1, *t*10, *t*11	3

4.1.7.3 Place invariants are generalized as shown in Table 4.10 when $M_0(p_{21}) = M_0(p_{22}) = M_0(p_{23}) = 1$.

Table 4.10. Generalized place invariants when one token is deposited in each shared resource place (p_{21}, p_{22}, and p_{23}).

$$\mu_3 + \mu_7 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_6 + \mu_7 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_3 + \mu_4 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_4 + \mu_6 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq (\mu_{21} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_6 + \mu_8 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{25} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$

4.1.7.4 When 8 monitors shown in Table 4.9 are added to the original *PNM1*, the optimally controlled *PNM1*, shown in Fig. 4.10, is obtained, which is live and can reach all 205 good states.

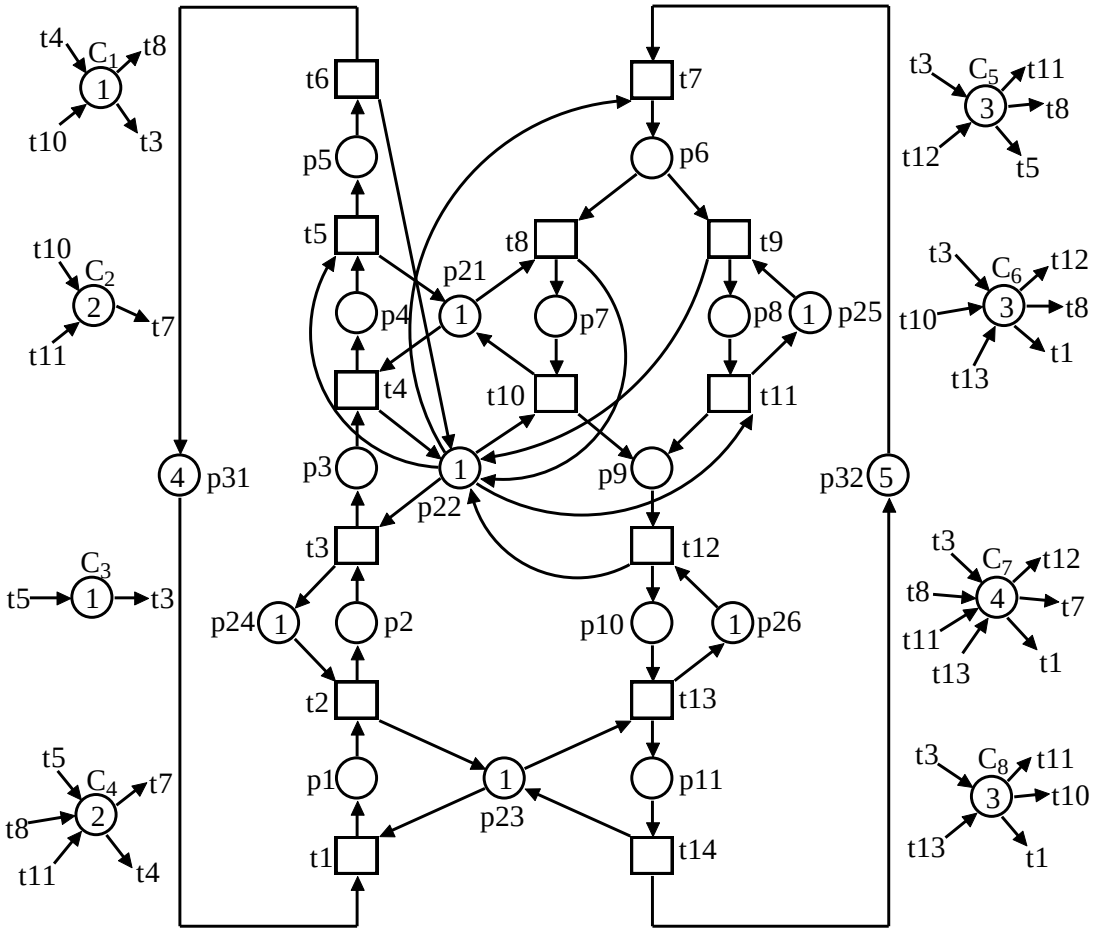

Figure 4.10. Optimally controlled *PNM1*.

4.1.8.1 The number of tokens in the adjustable shared resource places are set to 1 for $M_0(p_{21})$ and $M_0(p_{22})$ and set to 2 for $M_0(p_{23})$ as shown in Fig. 4.11.

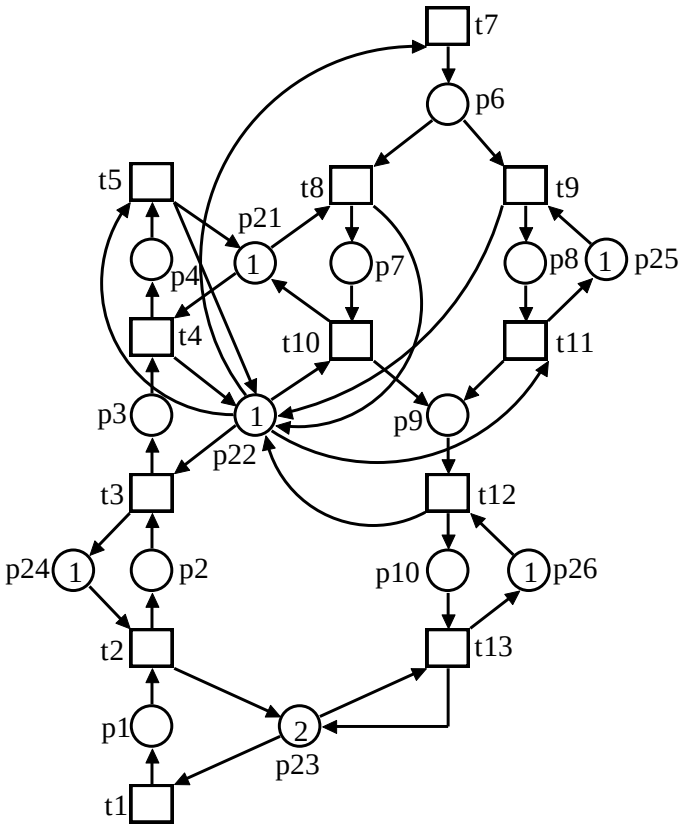

Figure 4.11. Reduced *PNM2* (*RPNM2*) with one token each is deposited in the shared resource places *p21* and *p22* and two tokens are deposited in the shared resource place *p23*.

4.1.8.2 8 necessary monitors shown in Table 4.11 are computed after 8 iterations for the *RPNM2* shown in Fig. 4.11.

Table 4.11. Monitors computed for the *RPNM2*.

FBM_i	PI_i	C_i	$^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
$\mu_3 = 1, \mu_7 = 1$	$\mu_3 + \mu_7 \leq 1$	*C1*	*t*4, *t*10	*t*3, *t*8	1
$\mu_3 = 1, \mu_4 = 1$	$\mu_3 + \mu_4 \leq 1$	*C2*	*t*5	*t*3	1
$\mu_6 = 1, \mu_7 = 1, \mu_8 = 1$	$\mu_6 + \mu_7 + \mu_8 \leq 2$	*C3*	*t*10, *t*11	*t*7	2
$\mu_4 = 1, \mu_6 = 1, \mu_8 = 1$	$\mu_4 + \mu_6 + \mu_8 \leq 2$	*C4*	*t*5, *t*8, *t*11	*t*4, *t*7	2
$\mu_1 = 2, \mu_2 = 1, \mu_7 = 1, \mu_9 = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq 4$	*C5*	*t*3, *t*12	*t*5, *t*8, t11	4
$\mu_1 = 2, \mu_2 = 1, \mu_7 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq 4$	*C6*	t3, t10, t13	t1, t8, *t*12	4
$\mu_1 = 2, \mu_2 = 1, \mu_9 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq 4$	*C7*	*t*3, *t*13	*t*1, *t*10, *t*11	4
$\mu_1 = 2, \mu_2 = 1, \mu_6 = 1, \mu_8 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_6 + \mu_8 + \mu_{10} \leq 5$	*C8*	*t*3, *t*8, *t*11, *t*13	*t*1, *t*7, *t*12	5

4.1.8.3 Place invariants are generalized as shown in Table 4.12 when $M_0(p_{21}) = M_0(p_{22}) = 1$ and $M_0(p_{23}) = 2$.

Table 4.12. Generalized place invariants when one token is deposited in the shared resource places p_{21} and p_{22} and two tokens are deposited in the shared resource place p_{23}.

$$\mu_3 + \mu_7 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_3 + \mu_4 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_6 + \mu_7 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_4 + \mu_6 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq (\mu_{21} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_6 + \mu_8 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{25} + \mu_{26}) - 1$$

4.1.8.4 When 8 monitors shown in Table 4.11 are added to the original *PNM2*, the optimally controlled *PNM2*, shown in Fig. 4.12, is obtained which is live and can reach all 421 good states.

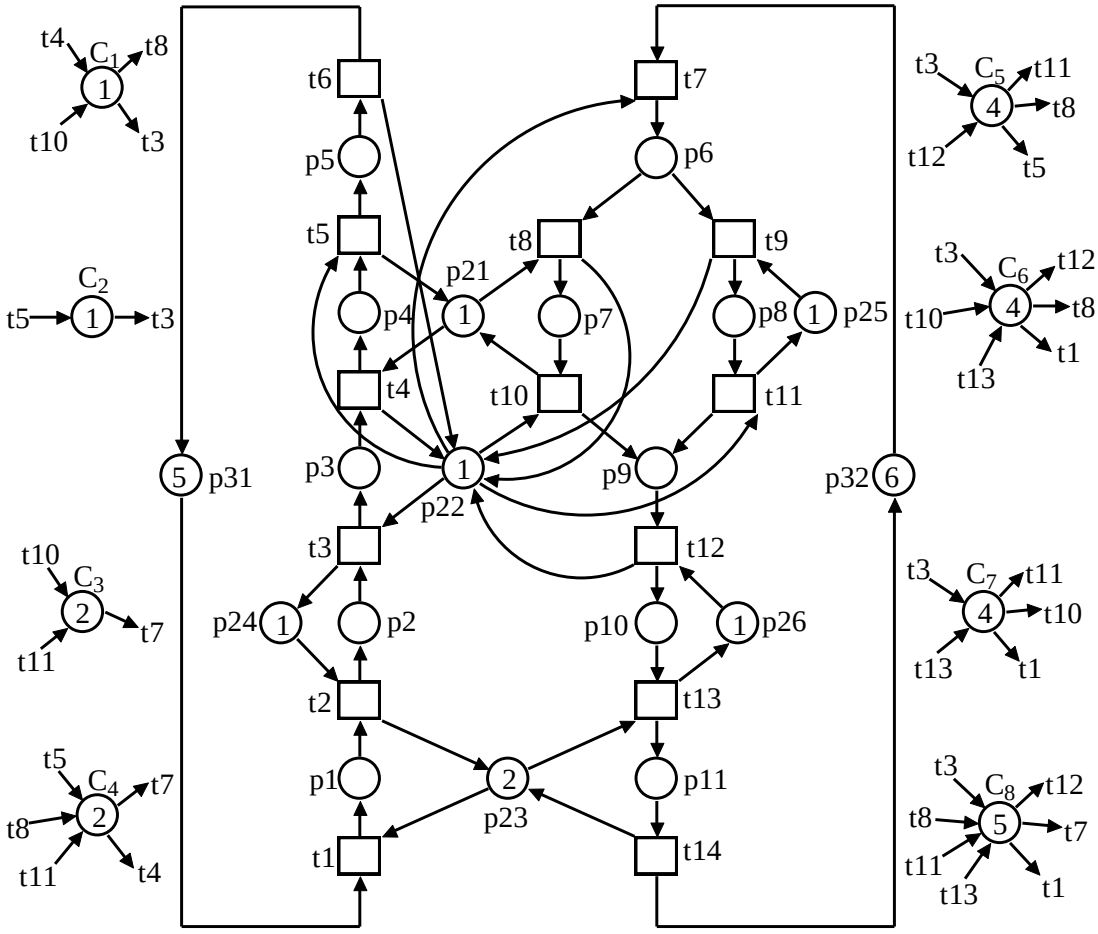

Figure 4.12. Optimally controlled *PNM2*.

4.1.9.1 The number of tokens in the adjustable shared resource places are set to 1 for $M_0(p_{21})$ and $M_0(p_{22})$ and set 3 for $M_0(p_{23})$ as shown in Fig. 4.13.

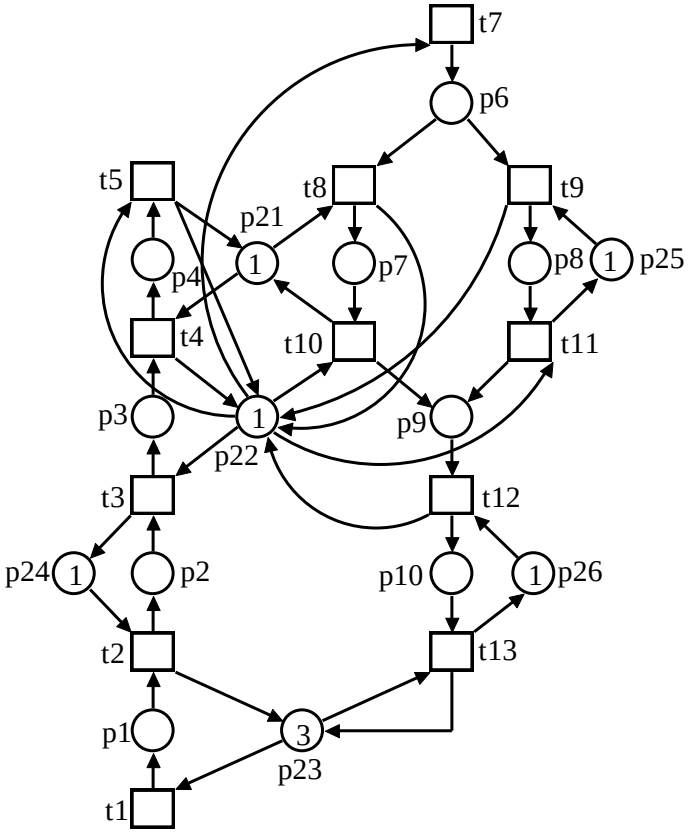

Figure 4.13. Reduced *PNM3* (*RPNM3*) with one token each is deposited in the shared resource places *p21* and *p22* and three tokens are deposited in the shared resource place *p23*

4.1.9.2 8 necessary monitors shown in Table 4.13 are computed after 8 iterations for the *RPNM 3* shown in Fig. 4.13.

Table 4.13. Monitors computed for the *RPNM3*.

FBM_i	PI_i	C_i	$^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
$\mu_3 = 1, \mu_7 = 1$	$\mu_3 + \mu_7 \leq 1$	*C1*	t4, *t*10	*t*3, *t*8	1
$\mu_3 = 1, \mu_4 = 1$	$\mu_3 + \mu_4 \leq 1$	*C2*	*t*5	*t*3	1
$\mu_6 = 1, \mu_7 = 1, \mu_8 = 1$	$\mu_6 + \mu_7 + \mu_8 \leq 2$	*C3*	*t*10, *t*11	*t*7	2
$\mu_4 = 1, \mu_6 = 1, \mu_8 = 1$	$\mu_4 + \mu_6 + \mu_8 \leq 2$	*C4*	*t*5, *t*8, *t*11	*t*4, *t*7	2
$\mu_1 = 3, \mu_2 = 1, \mu_7 = 1, \mu_9 = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq 5$	*C5*	*t*3, *t*12	*t*5, *t*8, *t*11	5
$\mu_1 = 3, \mu_2 = 1, \mu_7 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq 5$	*C6*	*t*3, *t*10, *t*13	*t*1, *t*8, *t*12	5
$\mu_1 = 3, \mu_2 = 1, \mu_9 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq 5$	*C7*	*t*3, *t*13	*t*1, *t*10, *t*11	5
$\mu_1 = 3, \mu_2 = 1, \mu_6 = 1, \mu_8 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_6 + \mu_8 + \mu_{10} \leq 6$	*C8*	*t*3, *t*8, *t*11, *t*13	*t*1, *t*7, *t*12	6

4.1.9.3 Place invariants are generalized as shown in Table 4.14 when $M_0(p_{21}) = M_0(p_{22}) = 1$ and $M_0(p_{23}) = 3$.

Table 4.14. Generalized place invariants when one token is deposited in the shared resource places p_{21} and p_{22} and three tokens are deposited in the shared resource place p_{23}.

$$\mu_3 + \mu_7 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_3 + \mu_4 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_6 + \mu_7 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_4 + \mu_6 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq (\mu_{21} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_6 + \mu_8 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{25} + \mu_{26}) - 1$$

4.1.9.4 When 8 monitors shown in Table 4.13 are added to the original *PNM*3, the optimally controlled *PNM*3, shown in Fig. 4.14, is obtained which is live and can reach all 709 good states.

4.2.1 DISCUSSION FOR THE SCENARIO 2

When there is only one token each in resource places *p21*, *p22* and *p23* the generalized *PIs* shown in Table 4.10 are valid. On the other hand, it is obvious that when there is one token deposited in each shared resource places *p21* and *p22*, and two or three tokens in the shared resource place *p23*, the generalized *PIs* are the same as depicted in Tables 4.12 and 4.14. This means that these generalized *PIs* are valid for the instances $M_0(p_{21}) = M_0(p_{22}) = 1$ while $M_0(p_{23}) = N$, $N = 2, 3, 4, \ldots$.

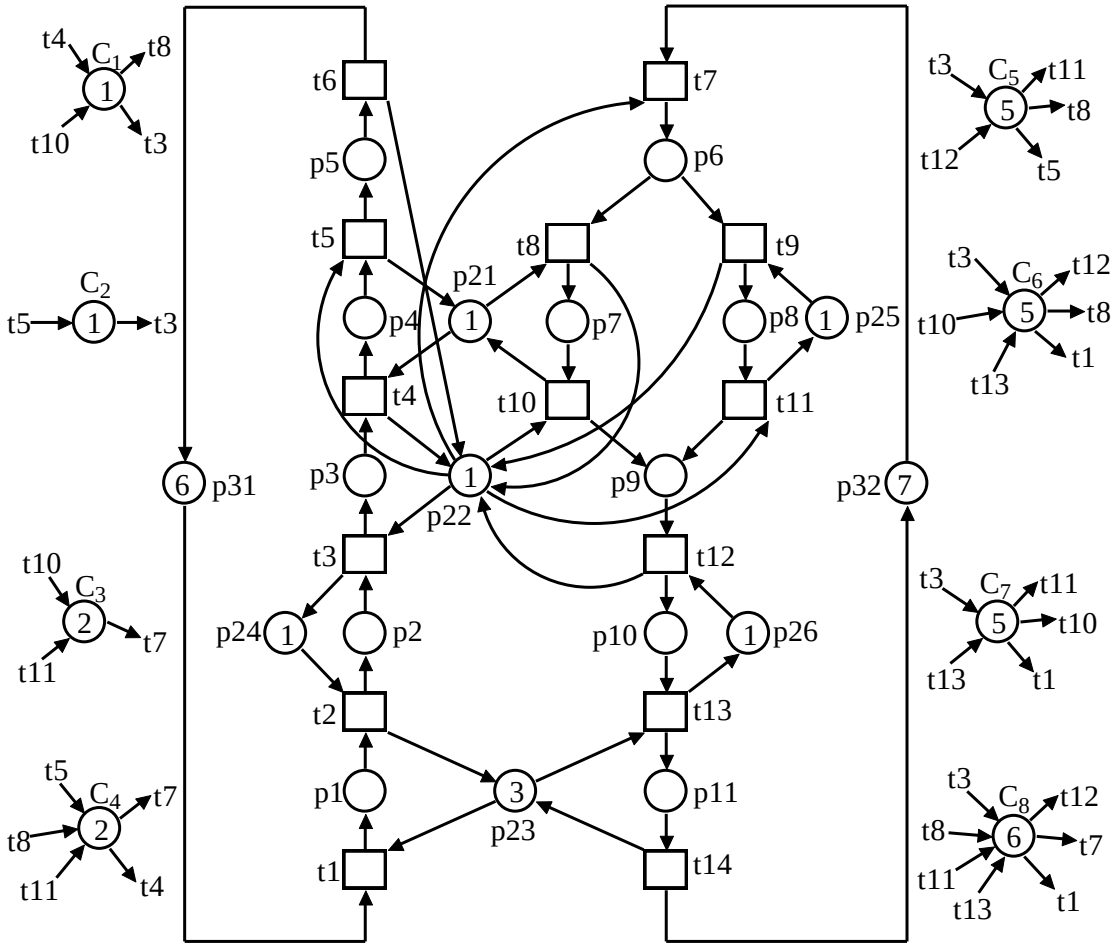

Figure 4.14. Optimally controlled *PNM3*.

4.2.2 REACHABILITY STATES COMPARISON BETWEEN THE ORIGINAL PNM AND THE REDUCED PNM FOR THE SCENARIO 2

Tables 4.15 and 4.16 present numerical experiments to compare the live states generated in the *RPNM* and the original *PNM* for different set of values of tokens in the shared resource places ($p_{21,}$ p_{22} and p_{23}). These results verify the correctness of generalized *PIs* obtained.

Table 4.15. Analysis results for the *RPNM* with different instances of number of tokens in the shared resource places.

Instance	Reduced net			Controlled net
(μ_{21}, μ_{22}, μ_{23})	N_{RG}	N_{DZ}	N_{LZ}	N_{RG}
(1,1,1)	176	57	119	119
(1,1,2)	264	81	183	183
(1,1,3)	352	105	247	247
(1,1,4)	440	129	311	311
(1,1,5)	528	153	375	375
(1,1,6)	616	177	439	439
(1,1,7)	704	201	503	503
(1,1,8)	792	225	567	567
(1,1,9)	880	249	631	631
(1,1,10)	968	273	695	695
(1,1,11)	1056	297	759	759

Table 4.16. Analysis results for the *PNM* with different instances of number of tokens in the shared resource places.

Instance	Original net			Controlled net
(μ_{21}, μ_{22}, μ_{23})	N_{RG}	N_{DZ}	N_{LZ}	N_{RG}
(1,1,1)	282	77	205	205
(1,1,2)	570	149	421	421
(1,1,3)	954	245	709	709
(1,1,4)	1434	365	1069	1069
(1,1,5)	2010	509	1501	1501
(1,1,6)	2682	677	2005	2005
(1,1,7)	3450	869	2581	2581
(1,1,8)	4314	1085	3229	3229
(1,1,9)	5274	1325	3949	3949
(1,1,10)	6330	1589	4741	4741
(1,1,11)	7482	1877	5605	5605

4.2.3 SCENARIO 3 (COMPUTATION OF MONITORS)

The computation of monitors are carried out based on the scenario 3: The number of tokens in the adjustable shared resource places are set as follows: $M_0(p_{22})$ = $M_0(p_{23})$ = 1 are always constant while $M_0(p_{21})$ = N, N = 1, 2, 3.

4.2.3.1 The number of tokens in the adjustable shared resource places are set to 1 ($M_0(p_{21})$ = $M_0(p_{22})$ = $M_0(p_{23})$ = 1) as shown in Fig. 4.15.

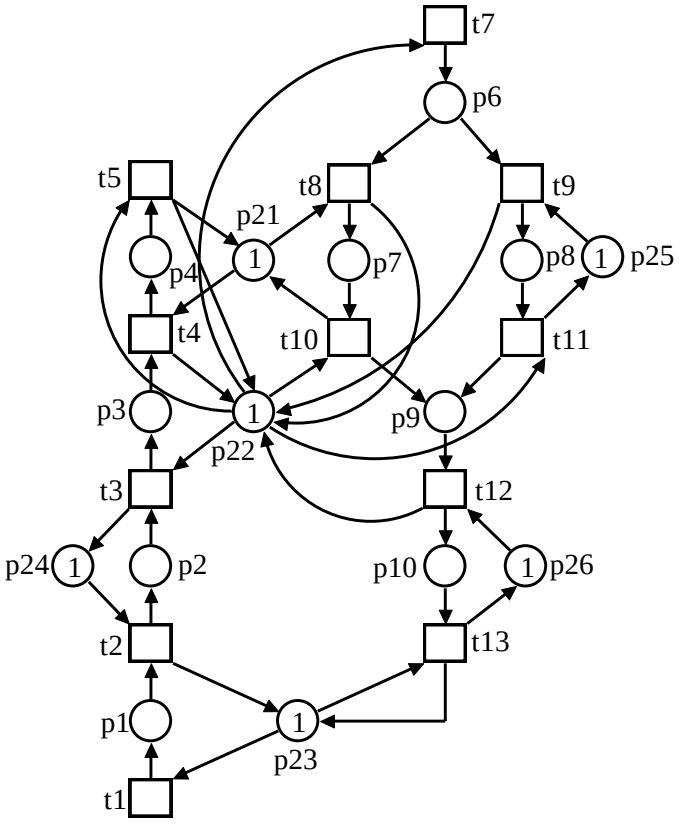

Figure 4.15. Reduced *PNM1* (*RPNM1*) with one token deposited in each shared resource place.

4.2.3.2 8 necessary monitors, shown in Table 4.17, are computed after 8 iterations for the *RPNM1* shown in Fig. 4.15.

Table 4.17. Monitors computed for the *RPNM1*.

FBM_i	PI_i	C_i	$^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
$\mu_3 = 1, \mu_7 = 1$	$\mu_3 + \mu_7 \leq 1$	*C1*	*t*4, *t*10	*t*3, *t*8	1
$\mu_6 = 1, \mu_7 = 1, \mu_8 = 1$	$\mu_6 + \mu_7 + \mu_8 \leq 2$	*C2*	*t*10, *t*11	*t*7	2
$\mu_3 = 1, \mu_4 = 1$	$\mu_3 + \mu_4 \leq 1$	*C3*	*t*5	*t*3	1
$\mu_4 = 1, \mu_6 = 1, \mu_8 = 1$	$\mu_4 + \mu_6 + \mu_8 \leq 2$	*C4*	*t*5, *t*8, *t*11	*t*4, *t*7	2
$\mu_1 = 1, \mu_2 = 1, \mu_7 = 1, \mu_9 = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq 3$	*C5*	*t*3, *t*12	*t*5, *t*8, *t*11	3
$\mu_1 = 1, \mu_2 = 1, \mu_7 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq 3$	*C6*	*t*3, *t*10, *t*13	*t*1, *t*8, *t*12	3
$\mu_1 = 1, \mu_2 = 1, \mu_6 = 1, \mu_8 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_6 + \mu_8 + \mu_{10} \leq 4$	*C7*	*t*3, *t*8, *t*11, *t*13	*t*1, *t*7, *t*12	4
$\mu_1 = 1, \mu_2 = 1, \mu_9 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq 3$	*C8*	*t*3, *t*13	*t*1, *t*10, *t*11	3

4.2.3.3 Place invariants are generalized as shown in Table 4.18 when $M_0(p_{21}) = M_0(p_{22}) = M_0(p_{23}) = 1$.

Table 4.18. Generalized place invariants when one token is deposited in each shared resource place (p_{21}, p_{22}, and p_{23}).

$$\mu_3 + \mu_7 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_6 + \mu_7 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_3 + \mu_4 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_4 + \mu_6 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq (\mu_{21} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_6 + \mu_8 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{25} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$

4.2.3.4 When 8 monitors shown in Table 4.17 are added to the original *PNM1*, the optimally controlled *PNM1*, shown in Fig. 4.16, is obtained which is live and can reach all 205 good states.

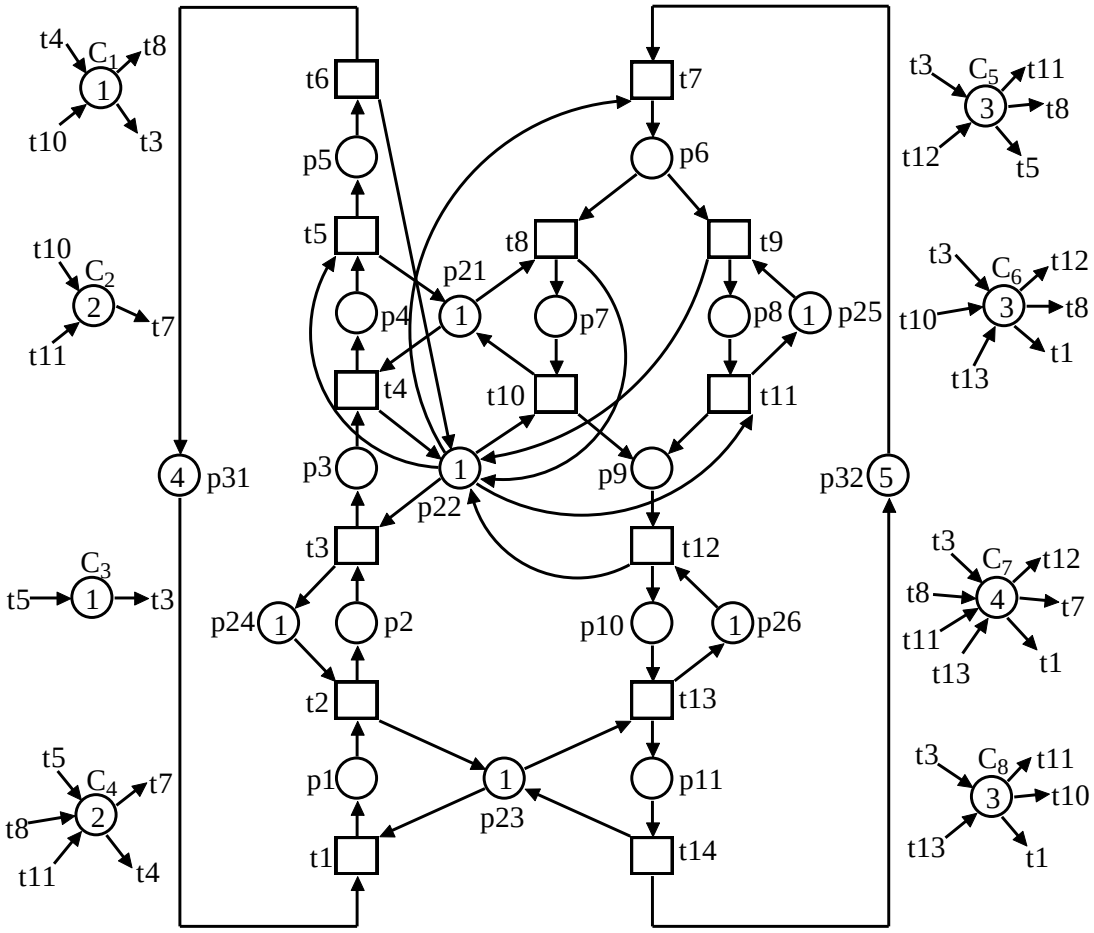

Figure 4.16. Optimally controlled *PNM1*.

4.2.4.1 The number of tokens in the adjustable shared resource places are set to 1 for $M_0(p_{22})$ and $M_0(p_{23})$ and set to 2 for $M_0(p_{21})$ as shown in Fig. 4.17.

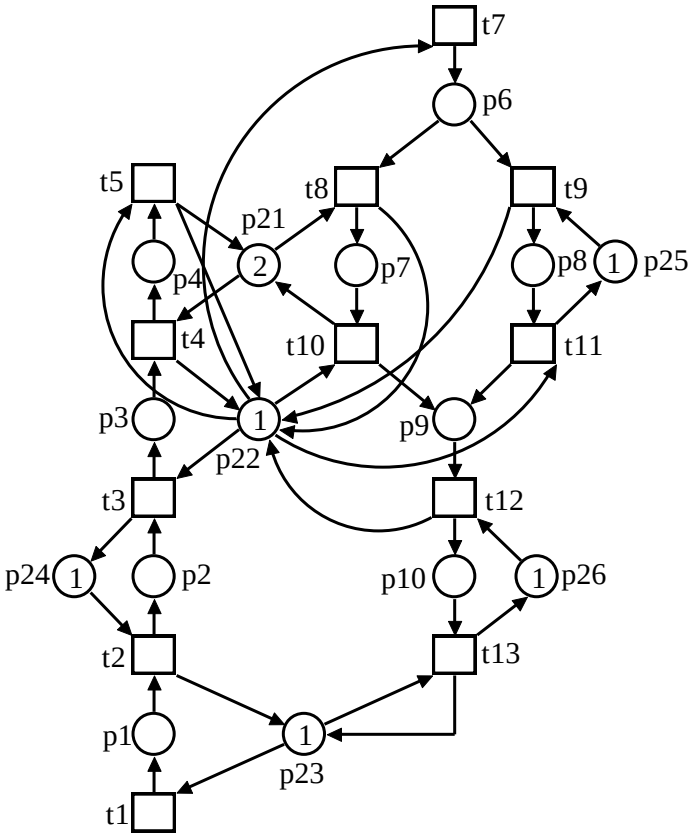

Figure 4.17. Reduced *PNM2* (*RPNM2*) with one token each is deposited in the shared resource places *p22* and *p23* and two tokens are deposited in the shared resource place *p21*.

4.2.4.2 6 necessary monitors, shown in Table 4.19, are computed after 8 iterations for the *RPNM2* shown in Fig. 4.17.

Table 4.19. Monitors computed for the *RPNM2*.

FBM_i	PI_i	C_i	$^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
$\mu_3 = 1$, $\mu_4 = 1$, $\mu_7 = 1$,	$\mu_3 + \mu_4 + \mu_7 \leq 2$	*C1*	*t*5, *t*10	*t*3, *t*8	2
$\mu_4 = 1$, $\mu_6 = 1$, $\mu_7 = 1$, $\mu_8 = 1$,	$\mu_4 + \mu_6 + \mu_7 + \mu_8 \leq 3$	*C2*	*t*5, *t*10, *t*11	*t*4, *t*7	3
$\mu_1 = 1$, $\mu_2 = 1$, $\mu_9 = 1$, $\mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq 3$	*C3*	*t*3, *t*13	*t*1, *t*10, *t*11	3
$\mu_1 = 1$, $\mu_2 = 1$, $\mu_7 = 2$, $\mu_9 = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq 4$	*C4*	*t*3, *t*12	*t*1, *t*8, *t*11	4
$\mu_1 = 1$, $\mu_2 = 1$, $\mu_7 = 2$, $\mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq 4$	*C5*	*t*3, *t*10, *t*13	*t*1, *t*8, *t*12	4
$\mu_1 = 1$, $\mu_2 = 1$, $\mu_6 = 1$, $\mu_7 = 1$, $\mu_8 = 1$, $\mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_6 + \mu_7 + \mu_8 + \mu_{10} \leq 5$	*C6*	*t*3, *t*10, *t*11, *t*13	*t*1, *t*7, *t*12	5

4.2.4.3 Place invariants are generalized as shown in Table 4.20 when $M_0(p_{22}) = M_0(p_{23}) = 1$ and $M_0(p_{21}) = 2$.

Table 4.20. Generalized place invariants when one token is deposited in the shared resource places p_{22} and p_{23} and two tokens are deposited in the shared resource place p_{21}.

$$\mu_3 + \mu_4 + \mu_7 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_4 + \mu_6 + \mu_7 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq (\mu_{21} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_6 + \mu_7 + \mu_8 + \mu_{10} \leq (\mu_{21} + \mu_{23} + \mu_{24} + \mu_{25} + \mu_{26}) - 1$$

4.2.4.4 When 6 monitors shown in Table 4.19 are added to the original *PNM2*, the optimally controlled *PNM2*, shown in Fig. 4.18, is obtained which is live and can reach all 484 good states.

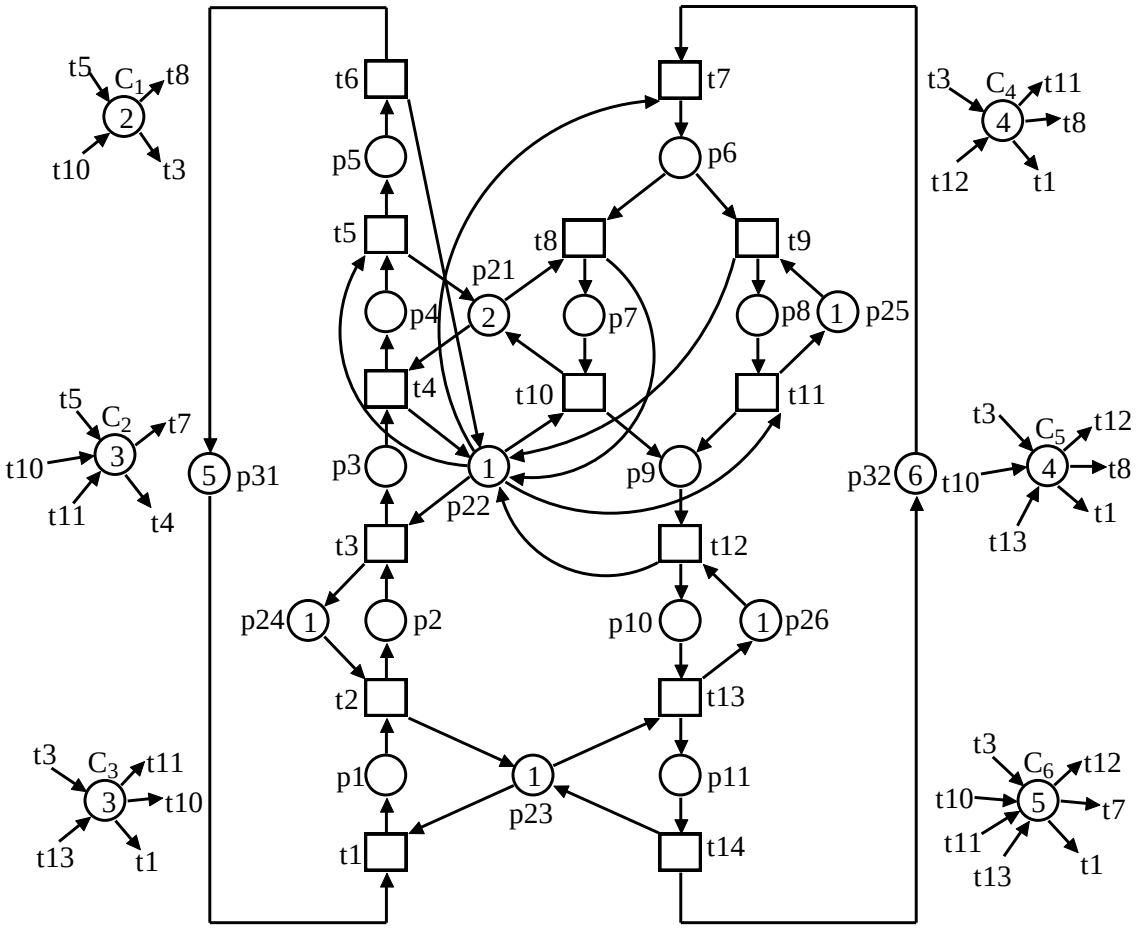

Figure 4.18. Optimally controlled *PNM2*.

4.2.5.1 The number of tokens in the adjustable shared resource places are set to 1 for $M_0(p_{22})$ and $M_0(p_{23})$ and set to 3 for $M_0(p_{21})$ as shown in Fig. 4.19.

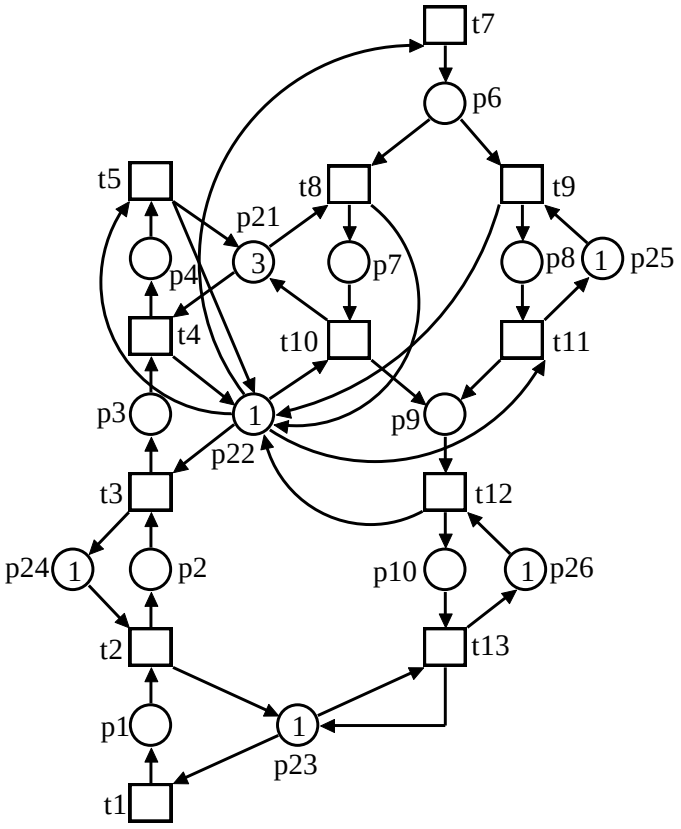

Figure 4.19. Reduced PNM*3* (*RPNM3*) with one token each is deposited in the shared resource places *p22* and *p23* and three tokens are deposited in the shared resource place *p21*

4.2.5.2 6 necessary monitors, shown in Table 4.21, are computed after 8 iterations for the *RPNM3* shown in Fig. 4.19.

Table 4.21. Monitors computed for the *RPNM3*.

FBM_i	PI_i	C_i	$^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
$\mu_3 = 1$, $\mu_4 = 1$, $\mu_7 = 2$,	$\mu_3 + \mu_4 + \mu_7 \leq 3$	*C1*	*t*5, *t*10	*t*3, *t*8	3
$\mu_4 = 1$, $\mu_6 = 1$, $\mu_7 = 2$, $\mu_8 = 1$,	$\mu_4 + \mu_6 + \mu_7 + \mu_8 \leq 4$	*C2*	*t*5, *t*10, *t*11	*t*4, *t*7	4
$\mu_1 = 1$, $\mu_2 = 1$, $\mu_9 = 1$, $\mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq 3$	*C3*	*t*3, *t*13	*t*1, *t*10, *t*11	3
$\mu_1 = 1$, $\mu_2 = 1$, $\mu_7 = 3$, $\mu_9 = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq 5$	*C4*	*t*3, *t*12	*t*1, *t*8, *t*11	5
$\mu_1 = 1$, $\mu_2 = 1$, $\mu_7 = 3$, $\mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq 5$	*C5*	*t*3, *t*10, *t*13	*t*1, *t*8, *t*12	5
$\mu_1 = 1$, $\mu_2 = 1$, $\mu_6 = 1$, $\mu_7 = 2$, $\mu_8 = 1$, $\mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_6 + \mu_7 + \mu_8 + \mu_{10} \leq 6$	*C6*	*t*3, *t*10, *t*11, *t*13	*t*1, *t*7, *t*12	6

4.2.5.3 Place invariants are generalized as shown in Table 4.22 when $M_0(p_{22}) = M_0(p_{23}) = 1$ and $M_0(p_{21}) = 3$.

Table 4.22. Generalized place invariants when one token is deposited in the shared resource place p_{22} and p_{23} and three tokens are deposited in the shared resource place p_{21}.

$$\mu_3 + \mu_4 + \mu_7 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_4 + \mu_6 + \mu_7 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq (\mu_{21} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_6 + \mu_7 + \mu_8 + \mu_{10} \leq (\mu_{21} + \mu_{23} + \mu_{24} + \mu_{25} + \mu_{26}) - 1$$

4.2.5.4. When 6 monitors shown in Table 4.21 are added to the original *PNM3*, the optimally controlled *PNM3*, shown in Fig. 4.20, is obtained which is live and can reach all 879 good states.

4.3.1 DISCUSSION FOR THE SCENARIO 3

When there is only one token each in shared resource places *p21*, *p22* and *p23*, the generalized *PIs* shown in Table 4.18 are valid. On the other hand, it is obvious that when there is one token deposited in each shared resource places *p22* and *p23*, and two or three tokens are deposited in the shared resource place *p21*, the generalized *PIs* are the same as depicted in Tables 4.20 and 4.22. This means that these generalized *PIs* are valid for the instances $M_0(p22) = M_0(p23) = 1$ while $M_0(p21) = N$, $N = 2, 3, 4, \ldots$.

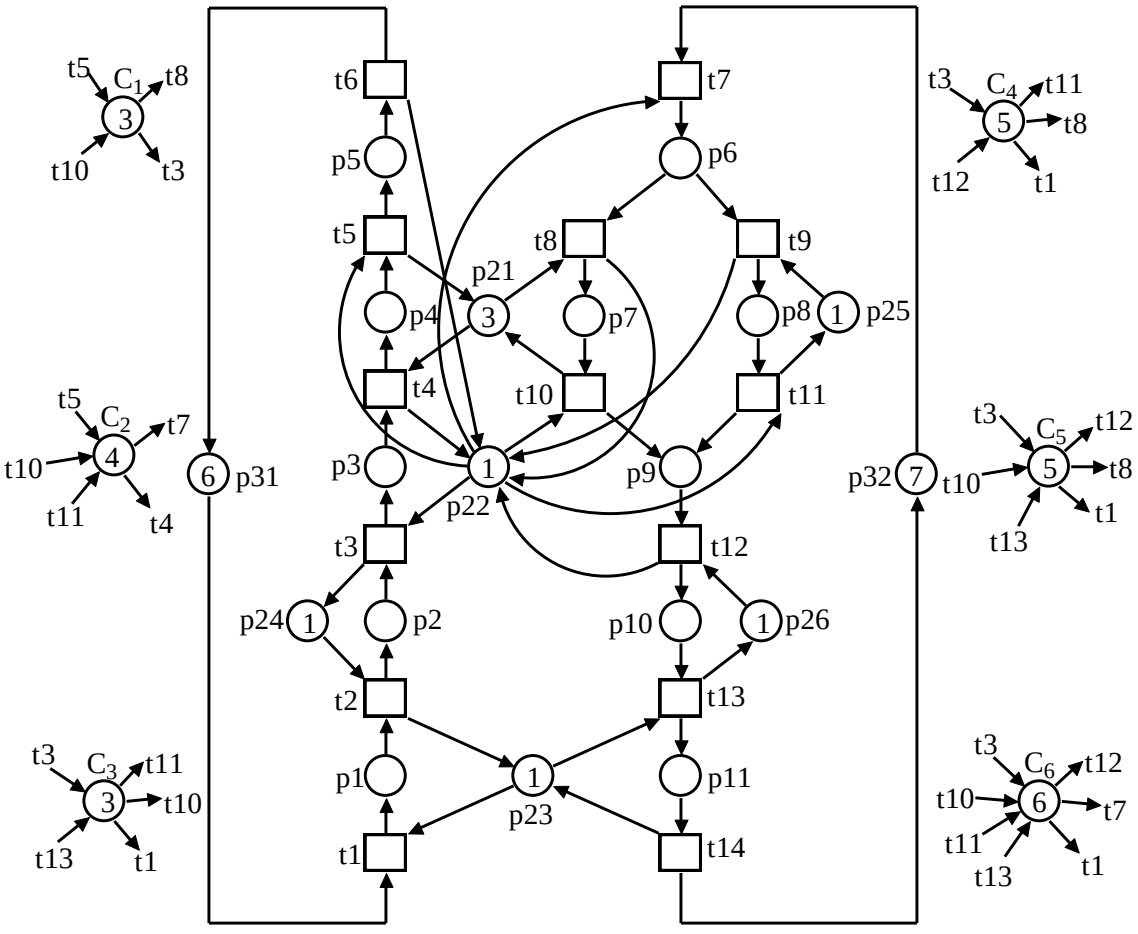

Figure 4.20. Optimally controlled *PNM3*.

4.3.2 REACHABILITY STATES COMPARISON BETWEEN THE ORIGINAL PNM AND THE REDUCED PNM FOR THE SCENARIO 3

Tables 4.23 and 4.24 present numerical experiments to compare the live states generated in the *RPNM* and the original *PNM* for different set of values of tokens in the shared resource places (p_{21}, p_{22} and p_{23}). These results verify the correctness of generalized *PIs* obtained.

Table 4.23. Analysis results for the *RPNM* with different instances of number of tokens in the shared resource places.

Instance	Reduced net			Controlled net
(μ_{21}, μ_{22}, μ_{23})	N_{RG}	N_{DZ}	N_{LZ}	N_{RG}
(1,1,1)	176	57	119	119
(2,1,1)	360	86	274	274
(3,1,1)	608	117	491	491
(4,1,1)	920	150	770	770
(5,1,1)	1296	185	1111	1111
(6,1,1)	1736	222	1514	1514
(7,1,1)	2240	261	1979	1979
(8,1,1)	2808	302	2506	2506
(9,1,1)	3440	345	3095	3095
(10,1,1)	4136	390	3746	3746

Table 4.24. Analysis results for the *PNM* with different instances of number of tokens in the shared resource places.

Instance	Original net			Controlled net
(μ_{21}, μ_{22}, μ_{23})	N_{RG}	N_{DZ}	N_{LZ}	N_{RG}
(1,1,1)	282	77	205	205
(2,1,1)	600	116	484	484
(3,1,1)	1036	157	879	879
(4,1,1)	1590	200	1390	1390
(5,1,1)	2262	245	2017	2017
(6,1,1)	3052	292	2760	2760
(7,1,1)	3960	341	3619	3619
(8,1,1)	4986	392	4594	4594
(9,1,1)	6130	445	5685	5685
(10,1,1)	7392	500	6892	6892

4.3.3 SCENARIO 4 (COMPUTATION OF MONITORS)

The computation of monitors are carried out based on the scenario 4: The number of tokens in the adjustable shared resource places are set as follows: $M_0(p_{21}) = M_0(p_{23}) = 1$ are always constant while $M_0(p_{22}) = N$, $N = 1, 2, 3$.

4.3.3.1 The number of tokens in the adjustable shared resource places are set to 1 ($M_0(p_{21}) = M_0(p_{22}) = M_0(p_{23}) = 1$) as shown in Fig. 4.21.

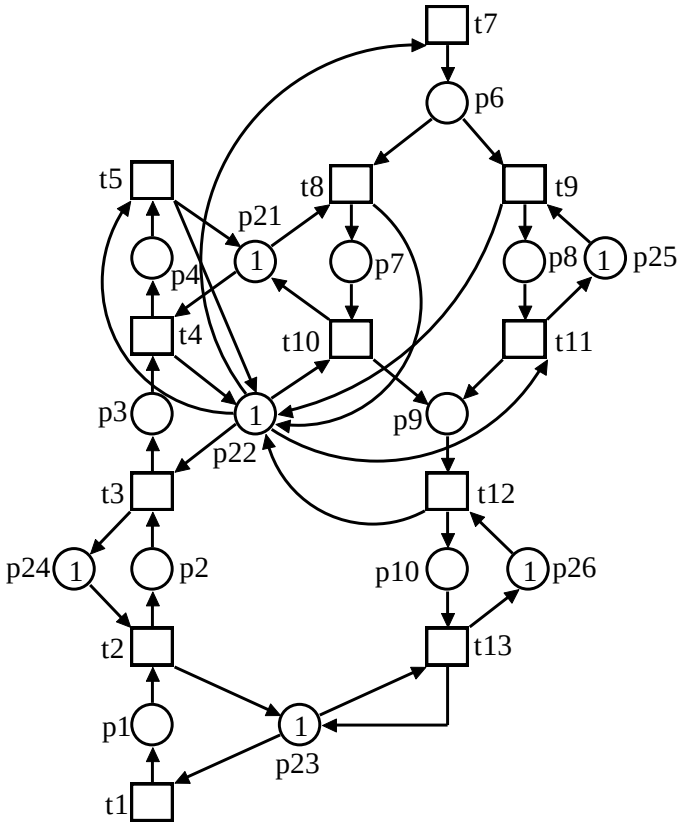

Figure 4.21. Reduced *PNM1* (*RPNM1*) with one token deposited in each shared resource place.

4.3.3.2 8 necessary monitors, shown in Table 4.25, are computed after 8 iterations for the *RPNM1* shown in Fig. 4.21.

Table 4.25. Monitors computed for the *RPNM1*.

FBM_i	PI_i	C_i	$^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
$\mu_3 = 1, \mu_7 = 1$	$\mu_3 + \mu_7 \leq 1$	*C1*	*t*4, *t*10	*t*3, *t*8	1
$\mu_6 = 1, \mu_7 = 1, \mu_8 = 1$	$\mu_6 + \mu_7 + \mu_8 \leq 2$	*C2*	*t*10, *t*11	*t*7	2
$\mu_3 = 1, \mu_4 = 1$	$\mu_3 + \mu_4 \leq 1$	*C3*	*t*5	*t*3	1
$\mu_4 = 1, \mu_6 = 1, \mu_8 = 1$	$\mu_4 + \mu_6 + \mu_8 \leq 2$	*C4*	*t*5, *t*8, *t*11	*t*4, *t*7	2
$\mu_1 = 1, \mu_2 = 1, \mu_7 = 1, \mu_9 = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq 3$	*C5*	*t*3, *t*12	*t*5, *t*8, *t*11	3
$\mu_1 = 1, \mu_2 = 1, \mu_7 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq 3$	*C6*	*t*3, *t*10, *t*13	*t*1, *t*8, *t*12	3
$\mu_1 = 1, \mu_2 = 1, \mu_6 = 1, \mu_8 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_6 + \mu_8 + \mu_{10} \leq 4$	*C7*	*t*3, *t*8, *t*11, *t*13	*t*1, *t*7, *t*12	4
$\mu_1 = 1, \mu_2 = 1, \mu_9 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq 3$	*C8*	*t*3, *t*13	*t*1, *t*10, *t*11	3

4.3.3.3 Place invariants are generalized as shown in Table 4.26 when $M_0(p_{21}) = M_0(p_{22}) = M_0(p_{23}) = 1$.

Table 4.26. Generalized place invariants when one token is deposited in each shared resource place (p_{21}, p_{22}, and p_{23}).

$$\mu_3 + \mu_7 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_6 + \mu_7 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_3 + \mu_4 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_4 + \mu_6 + \mu_8 \leq (\mu_{21} + \mu_{22} + \mu_{25}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_9 \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24}) - 1$$
$$\mu_1 + \mu_2 + \mu_7 + \mu_{10} \leq (\mu_{21} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_6 + \mu_8 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{25} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$

4.3.3.4 When 8 monitors shown in Table 4.25 are added to the original *PNM1*, the optimally controlled *PNM1*, shown in Fig. 4.22, is obtained which is live and can reach all 205 good states.

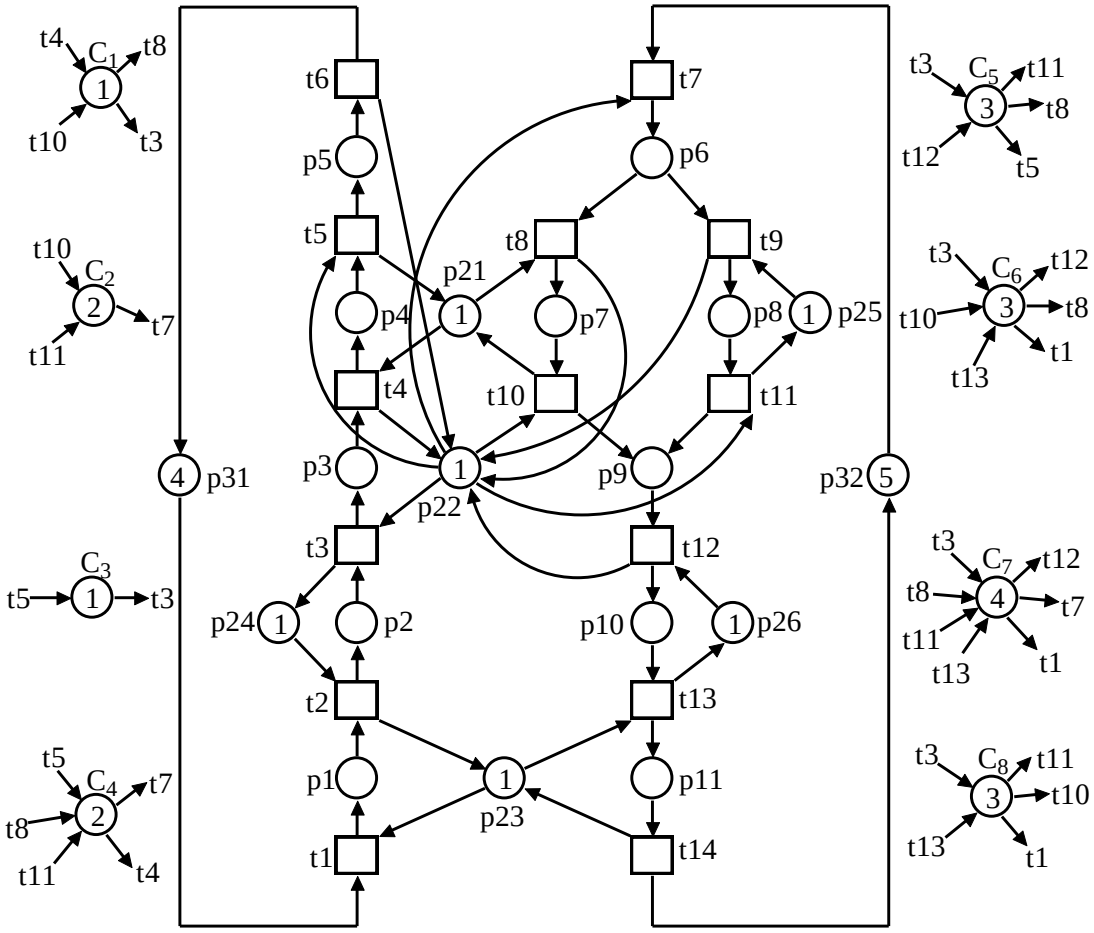

Figure 4.22. Optimally controlled *PNM1*.

4.3.4.1 The number of tokens in the adjustable shared resource places are set to 1 for $M_0(p_{21})$ and $M_0(p_{23})$ and set to 2 for $M_0(p_{22})$ as shown in Fig. 4.23.

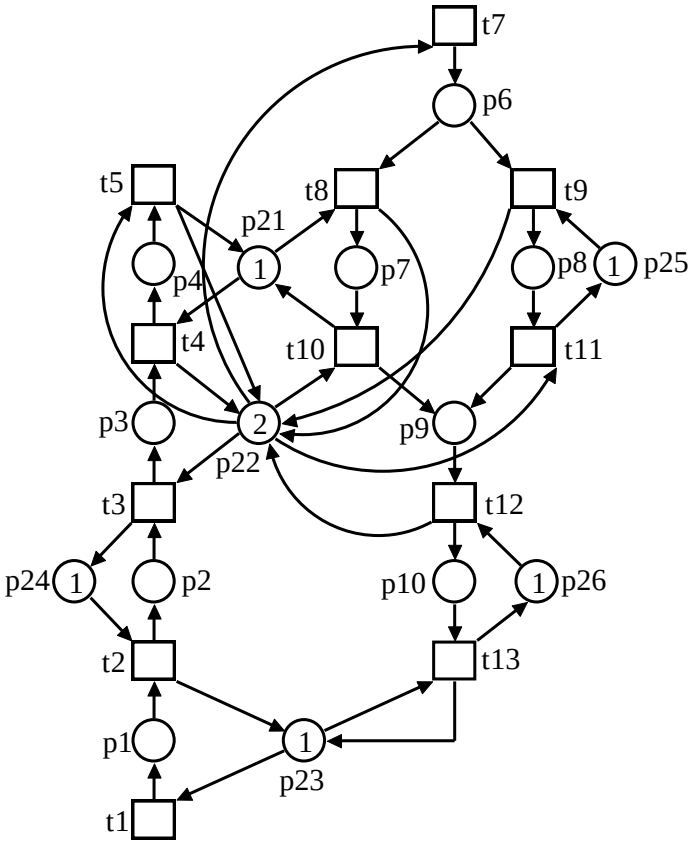

Figure 4.23. Reduced *PNM*2 (*RPNM2*) with one token each is deposited in the shared resource places *p21* and *p23* and two tokens are deposited in the shared resource place *p22*.

4.3.4.2 8 necessary monitors, shown in Table 4.27, are computed after 14 iterations for the *RPNM2* shown in Fig. 4.23.

Table 4.27. Monitors computed for the *RPNM2*.

FBM_i	PI_i	C_i	$^\bullet C_i$	$C_i^\bullet$	$\mu_0(C_i)$
$\mu_3 = 2, \mu_7 = 1$	$\mu_3 + \mu_7 \leq 2$	*C1*	*t*4, *t*10	*t*3, *t*8	2
$\mu_3 = 2, \mu_4 = 1$	$\mu_3 + \mu_4 \leq 2$	*C2*	*t*5	*t*3	2
$\mu_3 = 1, \mu_6 = 1, \mu_7 = 1, \mu_8 = 1$	$\mu_3 + \mu_6 + \mu_7 + \mu_8 \leq 3$	*C3*	*t*4, *t*10, *t*11	*t*3, *t*7	3
$\mu_3 = 1, \mu_4 = 1, \mu_6 = 1, \mu_8 = 1$	$\mu_3 + \mu_4 + \mu_6 + \mu_8 \leq 3$	*C4*	*t*5, *t*8, *t*11	*t*3, *t*7	3
$\mu_1 = 1, \mu_2 = 1, \mu_9 = 2, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq 4$	*C5*	*t*3, *t*13	*t*1, *t*10, *t*11	4
$\mu_1 = 1, \mu_2 = 1, \mu_3 = 1, \mu_7 = 1, \mu_9 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_3 + \mu_7 + \mu_9 + \mu_{10} \leq 5$	*C6*	*t*4, *t*13	*t*1, *t*8, *t*11	5
$\mu_1 = 1, \mu_2 = 1, \mu_3 = 1, \mu_4 = 1, \mu_9 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_9 + \mu_{10} \leq 5$	*C7*	*t*5, *t*13	*t*1, *t*10, *t*11	5
$\mu_1 = 1, \mu_2 = 1, \mu_4 = 1, \mu_6 = 1, \mu_8 = 1, \mu_9 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_4 + \mu_6 + \mu_8 + \mu_9 + \mu_{10} \leq 6$	*C8*	*t*3, *t*5, *t*13	*t*1, *t*4, *t*7	6

4.3.4.3 Place invariants are generalized as shown in Table 4.28 when $M_0(p_{21}) = M_0(p_{23}) = 1$ and $M_0(p_{22}) = 2$.

Table 4.28. Generalized place invariants when one token is deposited in the shared resource places p_{21} and p_{23} and two tokens are deposited in the shared resource place p_{22}.

$\mu_3 + \mu_7 \leq (\mu_{21} + \mu_{22}) - 1$
$\mu_4 + \mu_3 \leq (\mu_{21} + \mu_{22}) - 1$
$\mu_3 + \mu_6 + \mu_7 + \mu_8 \leq (\mu_{22} + \mu_{21} + \mu_{25}) - 1$
$\mu_3 + \mu_4 + \mu_6 + \mu_8 \leq (\mu_{22} + \mu_{21} + \mu_{25}) - 1$
$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$
$\mu_1 + \mu_2 + \mu_3 + \mu_7 + \mu_9 + \mu_{10} \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$
$\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_9 + \mu_{10} \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$
$\mu_1 + \mu_2 + \mu_4 + \mu_6 + \mu_8 + \mu_9 + \mu_{10} \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24} + \mu_{25} + \mu_{26}) - 1$

4.3.4.4 When 8 monitors shown in Table 4.27 are added to the original *PNM2*, the optimally controlled *PNM2*, shown in Fig. 4.24, is obtained which is live and can reach all 870 good states.

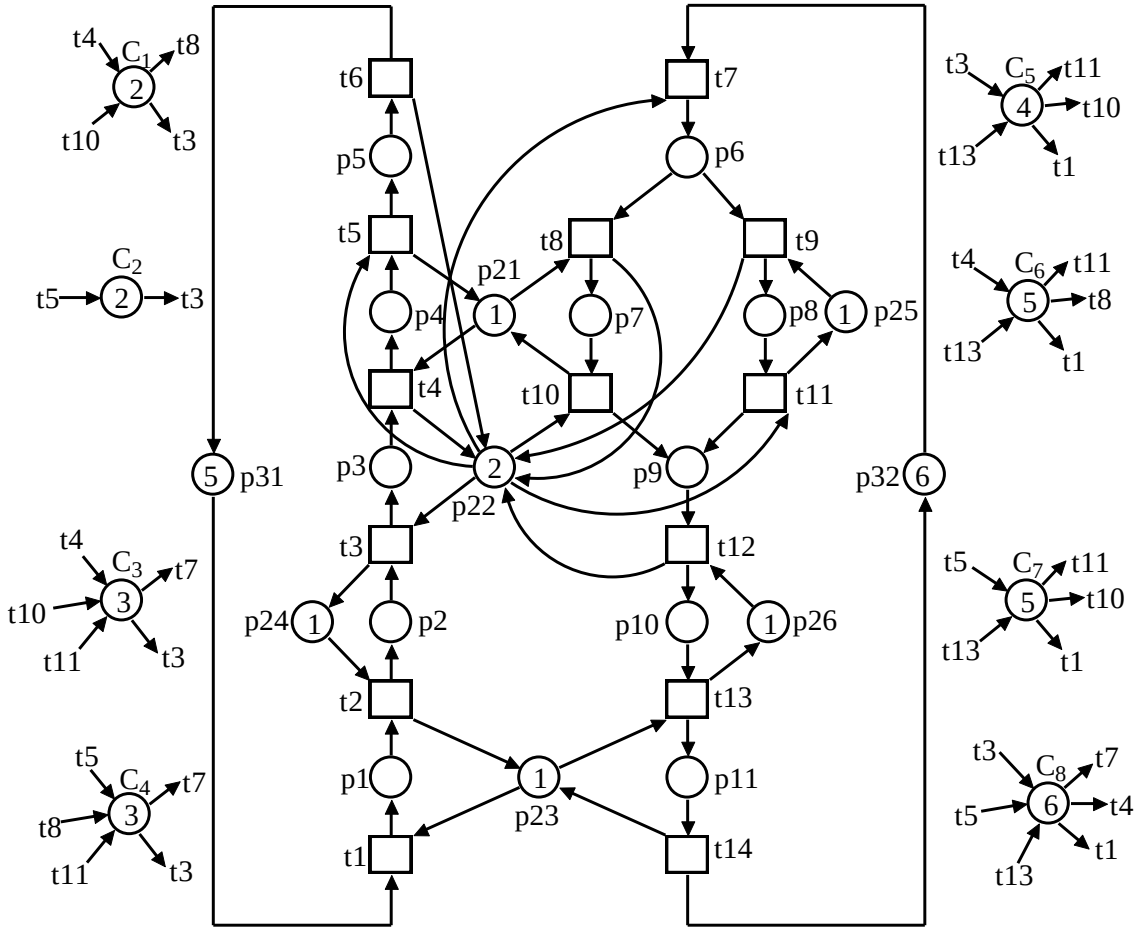

Figure 4.24. Optimally controlled *PNM2*.

4.3.5.1 The number of tokens in the adjustable shared resource places are set to 1 for $M_0(p_{21})$ and $M_0(p_{23})$ and set to 3 for $M_0(p_{22})$ as shown in Fig. 4.25.

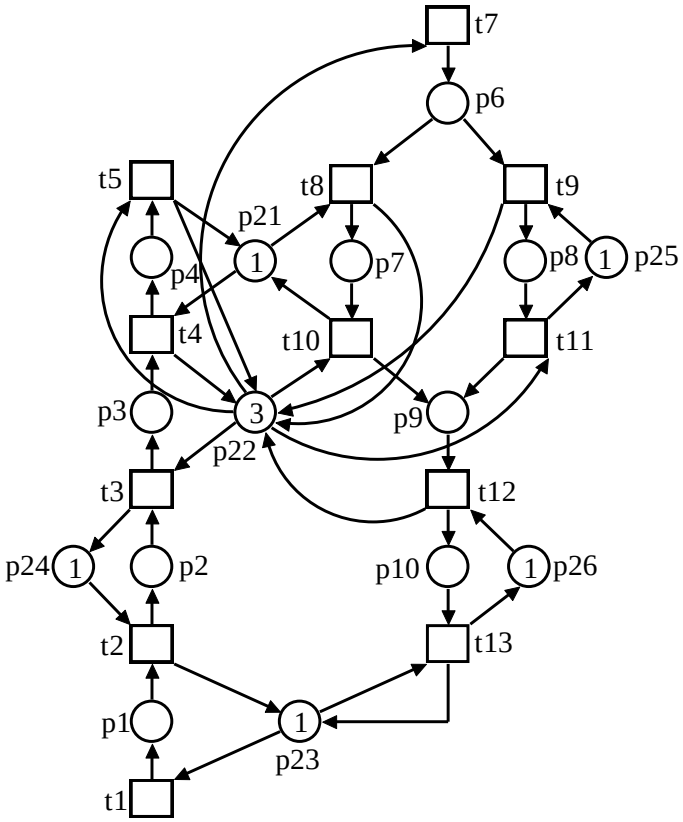

Figure 4.25. Reduced *PNM*3 (*RPNM3*) with one token each is deposited in the shared resource places *p21* and *p23* and three tokens are deposited in the shared resource place *p22*

4.3.5.2 8 necessary monitors, shown in Table 4.29, are computed after 14 iterations for the *RPNM3* shown in Fig. 4.25.

Table 4.29. Monitors computed for the *RPNM3*.

FBM_i	PI_i	C_i	$^{\bullet}C_i$	$C_i^{\bullet}$	$\mu_0(C_i)$
$\mu_3 = 3, \mu_7 = 1$	$\mu_3 + \mu_7 \leq 3$	*C1*	*t*4, *t*10	*t*3, *t*8	3
$\mu_3 = 3, \mu_4 = 1$	$\mu_3 + \mu_4 \leq 3$	*C2*	*t*5	*t*3	3
$\mu_3 = 1, \mu_6 = 2,$ $\mu_7 = 1, \mu_8 = 1$	$\mu_3 + \mu_6 + \mu_7 + \mu_8 \leq 4$	*C3*	*t*4, *t*10, *t*11	*t*3, *t*7	4
$\mu_3 = 1, \mu_4 = 1,$ $\mu_6 = 2, \mu_8 = 1$	$\mu_3 + \mu_4 + \mu_6 + \mu_8 \leq 4$	*C4*	*t*5, *t*8, *t*11	*t*3, *t*7	4
$\mu_1 = 1, \mu_2 = 1,$ $\mu_9 = 3, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq 5$	*C5*	*t*3, *t*13	*t*1, *t*10, *t*11	5
$\mu_1 = 1, \mu_2 = 1,$ $\mu_3 = 2, \mu_7 = 1,$ $\mu_9 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_3 + \mu_7 + \mu_9 + \mu_{10} \leq 6$	*C6*	*t*4, *t*13	*t*1, *t*8, *t*11	6
$\mu_1 = 1, \mu_2 = 1,$ $\mu_3 = 2, \mu_4 = 1,$ $\mu_9 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_9 + \mu_{10} \leq 6$	*C7*	*t*5, *t*13	*t*1, *t*10, *t*11	6
$\mu_1 = 1, \mu_2 = 1,$ $\mu_3 = 1, \mu_4 = 1,$ $\mu_6 = 1, \mu_8 = 1,$ $\mu_9 = 1, \mu_{10} = 1$	$\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_6 + \mu_8 + \mu_9 + \mu_{10} \leq 7$	*C8*	*t*5, *t*13	*t*1, *t*7	7

4.3.5.3 Place invariants are generalized as shown in Table 4.30 when $M_0(p_{21}) = M_0(p_{23}) = 1$ and $M_0(p_{22}) = 3$.

Table 4.30. Generalized place invariants when one token is deposited in the shared resource places p_{21} and p_{23} and three tokens are deposited in the shared resource place p_{22}.

$$\mu_3 + \mu_7 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_4 + \mu_3 \leq (\mu_{21} + \mu_{22}) - 1$$
$$\mu_3 + \mu_6 + \mu_7 + \mu_8 \leq (\mu_{22} + \mu_{21} + \mu_{25}) - 1$$
$$\mu_3 + \mu_4 + \mu_6 + \mu_8 \leq (\mu_{22} + \mu_{21} + \mu_{25}) - 1$$
$$\mu_1 + \mu_2 + \mu_9 + \mu_{10} \leq (\mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_3 + \mu_7 + \mu_9 + \mu_{10} \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_9 + \mu_{10} \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24} + \mu_{26}) - 1$$
$$\mu_1 + \mu_2 + \mu_3 + \mu_4 + \mu_6 + \mu_8 + \mu_9 + \mu_{10} \leq (\mu_{21} + \mu_{22} + \mu_{23} + \mu_{24} + \mu_{25} + \mu_{26}) - 1$$

4.3.5.4 When 8 monitors shown in Table 4.29 are added to the original *PNM3*, the optimally controlled *PNM3*, shown in Fig. 4.26, is obtained which is live and can reach all 2246 good states.

4.4.1 DISCUSSION FOR THE SCENARIO 4

When there is only one token each in shared resource places *p21*, *p22* and *p23*, the generalized *PIs* shown in Table 4.26 are valid, also when there is one token each in the shared resource places *p21* and *p23* and two tokens in the shared resource place *p22*, the generalized *PIs* shown in Table 4.28 are valid. On the other hand, it is obvious that when there is one token each in the shared resource places *p21* and *p22*, and three or four tokens in the shared resource place *p23*, the generalized *PIs* are the same as depicted in Table 4.30. This means that these generalized *PIs* are valid for the instances $M_0(p_{21}) = M_0(p_{23}) = 1$ while $M_0(p_{22}) = N$, $N = 3, 4, 5, \ldots$.

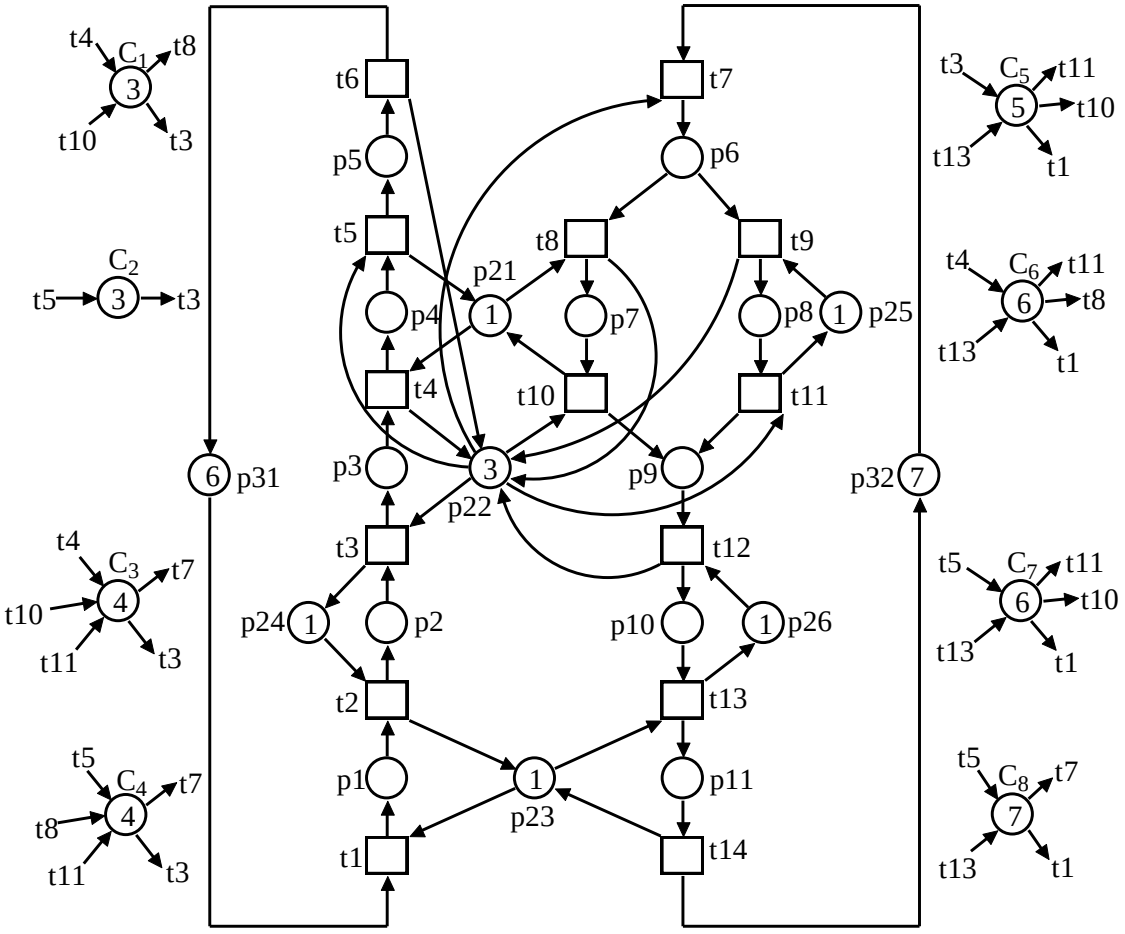

Figure 4.26. Optimally controlled *PNM3*.

4.4.2 REACHABILITY STATES COMPARISON BETWEEN THE ORIGINAL PNM AND THE REDUCED PNM FOR THE SCENARIO 4

Tables 4.31 and 4.32 present numerical experiments to compare the live states generated in the *RPNM* and the original *PNM* for different set of values of tokens in the shared resource places (p_{21}, p_{22} and p_{23}). These results verify the correctness of generalized *PIs* obtained.

Table 4.31. Analysis results for the *RPNM* with different instances of number of tokens in the shared resource places.

Instance (μ_{21}, μ_{22}, μ_{23})	Reduced net			Controlled net
	N_{RG}	N_{DZ}	N_{LZ}	N_{RG}
(1,1,1)	176	57	119	119
(1,2,1)	464	74	390	390
(1,3,1)	944	98	846	846
(1,4,1)	1664	129	1540	1540
(1,5,1)	2672	152	2520	2520
(1,6,1)	4016	182	3834	3834
(1,7,1)	5744	214	5530	5530
(1,8,1)	7904	248	7656	7656
(1,9,1)	10544	284	10260	10260
(1,10,1)	13712	322	13390	13390

Table 4.32. Analysis results for the *PNM* with different instances of number of tokens in the shared resource places.

Instance (μ_{21}, μ_{22}, μ_{23})	Original net			Controlled net
	N_{RG}	N_{DZ}	N_{LZ}	N_{RG}
(1,1,1)	282	77	205	205
(1,2,1)	972	102	870	870
(1,3,1)	2380	134	2246	2246
(1,4,1)	4866	168	4698	4698
(1,5,1)	8862	204	8658	8658
(1,6,1)	14872	242	14630	14630
(1,7,1)	23472	282	23190	23190
(1,8,1)	35310	324	34986	34986
(1,9,1)	51106	368	50738	50738
(1,10,1)	71652	414	71238	71238
(1,11,1)	97812	462	97350	97350

CHAPTER 5

CONCLUSION

5.1 THESIS CONCLUSION

In this study we consider the computation of Petri net based liveness-enforcing supervisors for flexible manufacturing systems (*FMS*) suffering from deadlock problems. As Petri nets become larger, their reachability graph (*RG*) grows exponentially due to the size of the Petri nets. This problem is called the state explosion problem. The solution to this problem is not easy with the currently available methods. In this study a new method is proposed to solve the computational cost problems in the design of *PN* based liveness-enforcing supervisors of *FMS*.

Studies on complicated Petri nets revealed that, the proposed method is very effective in dealing with the deadlock problems in *FMS*.

REFERENCES

[1] Christos G. Cassndras (Boston University) and Stephane Lafortune, "Introduction to Discrete Event Systems", Second Edition, pp. 52, Springer, 2008.

[2] M. C. Zhou, F. Dicesare, and D. Rudolph, "Design and Implementation of a Petri Net Based Supersor for a Flexible Manufacturing System", Autom. vol. 28, no. 6, pp. 1199-1208, 1992.

[3] N. Viswanadham and Y. Narahari, "Performance Modelling of Automated Manufacturing Systems", Englewood Cliffs, NJ: Pretence Hall, 1992.

[4] B. H. Krogh and Z. Banaszak, "Deadlok avoidance in Flexible Manufacturing System with Concurrently Competing Process Flows", IEEE Trans. Robot. Autom., vol. 6, no.6, pp. 724-173, 1990.

[5] N. Viswanadham, T. Johnson and Y. Narahari, "Deadlock Prevention and Deadlock Avoidance in Flexible Manufacturing Systems using Petri Net Models", IEEE Trans. Robot, Autom. vol. 6, no. 6, pp. 713-723, Dec. 1990.

[6] A. Wysk, H. Cho, T. Kumaran and W. Chang, "A Structured Approach to Deadlock Detection, Avoidance and Resolution in Flexible Manufacturing Systems", Int. J. Proc. Res., vol. 32, no. 10, pp. 2361-2379, 1994.

[7] M. C. Zhou, N. Q. Wu and Z. W. Li, "Deadlock Control of Automated Manufacturing Baesd on Petri Nets", IEEE Trans. on Syst., Man, Cybern., Part C. Application and Reviews, vol. 42, no. 4, pp. 437-462, July 2012.

[8] S. A. Reveliotis, "On the Siphon-Based Characterization of Liveness in Sequential Resource Allocation Systems", in Proc. Int. Conf. Appl. Theory Petri Nets, W. M. P. Vander Aalst and E. Best, Eds. Berlin, Germany: Springer-Verlag, vol. 2679, pp. 241-255, 2003.

[9] M. C. Zhou and Z. W. Li, "On Siphon Computation for Deadlock Control in a Class of Petri Nets", IEEE Trans. Syst., Man, Cybern., A, Syst., Humans, vol. 38, pp. 667-679, May, 2008.

[10] J. Zhang, M. C. Zhou and Z. W. Li, "Liveness-Enforcing Supervisor Design for a Class of Generalized Petri Net Models of Flexible Manufacturing Systems", IET Control Theory and Appl. vol. 4, no. 1, pp. 955-967, 2007.

[11] J. Ezpeleta, J. M. Colon and J. Martinez, "A Petri Net Based Deadlock Prevention Policy for Flexible Manufacturing Systems", IEEE Trans. Robot. Autom., vol. 11, no. 2, pp. 173-184, Apr. 1995.

[12] T. Murata, "Petri nets: Properties, Analysis, and Applications", Proc. IEEE, vol. 77, no. 4, pp. 541-580, Apr. 1989.

[13] Kwang-HyungLee, "Hierarchical Reduction Method for Analysis and Decomposition of Petri Nets", IEEE Trans. on Syst., Man, Cybern., vol. smc-15, no. 2, Mar/Apr. 1985.

[14] A. P. Sage, "Methodology for Large Scale Systems", New York:McGraw-Hill,198-5.

[15] J. N. Warfield, "Societal Systems: Planning, Policy and Complexity", New York: John Wiley, 1976.

[16] K. H. Lee and J. Favrel, "On Hierarchial Reduction Method for Analysis and Decomposition of Petri Nets", IEEE Trans. Syst., Man, Cybern., vol. 15, pp. 272-280, 1985.

[17] M. D. Jeng and M. C. Zhou, "Introduction to the Special Section on Petri Nets in Semiconductor Manufacturing", IEEE Trans. on Semiconductor Manufacturing, vol. 11, no. 3, pp. 330-332, Aug. 1998.

[18] M. Uzam and M.C. Zhou, "An Improved Iterative Synthesis Method for Liveness Enforcing Supervisors of Flexible Manufacturing Systems", Int. J. Prod. Res., vol. 44, Issue 10, pp. 1987-2030, 2006.

[19] M. Uzam, "The Use of Petri Net Reduction Approach for an Optimal Deadlock prevention Policy for Flexible Manufacturing Systems", Int. J. Adv. Manuf. Tech., vol. 23, no. (3-4), pp. 204-219, 2004.

[20] K. Yamalidou, J. Moody, M. Lemmon and P. Antsaklis, "Feedback Control of Petri Nets Based on Place Invariants Automatica", vol. 32, no. 1, pp. 15-28, 1996.

[21] M. Uzam and M.C. Zhou, "An Iterative Synthesis Approach to Petri Net Based Deadlock Prevention Policy for Flexible Manufacturing Systems", IEEE Trans. Syst., Man, Cybern., A. 2005, Accepted for Publication.

[22] PN-Tools, "A Petri Net Analysis Tool", Ver. 1.0, Pedagogical University of Rzesow Poland, 1987-1996.

[23] M. Uzam and M.C. Zhou, "An Iterative Synthesis Approach to Petri Net Based Deadlock Prevention Policy for Flexible Manufacturing Systems", in Proceedings of the 2004 Int. Conf. Syst., IEEE Trans. Syst., Man, Cybern,. The Haque, Netherlands, 10-13 Oct., pp. 4260-4265, 2004.

[24] M. Uzam, Z. W. Li, and M. C. Zhou, (2007), "Identification and Elimination of Redundant Control Places in Petri net Based Liveness Enforcing Supervisors of FMS", Int. J. Adv, Manuf. Technol., vol. 35, no. 1-2, pp. 150-167, 2007.