Sigma J Eng Nat Sci, Vol. 41, No. 3, pp. 493–502, June, 2023

Sigma Journal of Engineering and Natural Sciences
Web page info: https://sigma.yildiz.edu.tr
DOI: 10.14744/sigma.2023.00056

Research Article

Computation of the number of legal states for petri net-based deadlock prevention problems

Gökhan GELEN[1,*], Murat UZAM[2]

[1]Bursa Technical University, Department of Mechatronics Engineering, Bursa, 16310, Türkiye
[2]Yozgat Bozok University, Department of Electrical and Electronics Engineering, Yozgat, 66000, Türkiye

ARTICLE INFO

Article history
Received: 09 May 2021
Revised: 31 July 2021
Accepted: 15 September 2021

Keywords:
Discrete Event Systems; Petri Nets; Deadlock Prevention; Reachability Graph; Strongly Connected Components

ABSTRACT

Petri net (PN) based prevention and control methods are widely studied in the literature to solve deadlock problems in flexible manufacturing systems (FMS). In PN models of FMS suffering from deadlocks, the reachability graph (RG) of the PN model can provide all reachable system states from the initial state, including all legal states, bad states, and deadlock states. A maximally permissive deadlock controller allows the system to reach all legal states exist within the live zone (LZ) that determines the optimal live behavior, while prohibits reaching bad and deadlock states exist within the dead zone. It is necessary to know the exact number of legal states that must be provided by a deadlock controller to determine the behavioral permissiveness of a control policy. Therefore, the number of legal states has been considered as a quality measure for deadlock prevention methods available in the literature. Unfortunately, to date for a given RG of a PN model of an FMS suffering from deadlocks, no study has been reported to provide the number of reachable legal states exist within the LZ of the given RG. In this paper, a method is proposed for the computation of the number of legal states that must be provided by an optimal deadlock prevention policy. The proposed method makes use of the reachability analysis of a given PN model of a deadlock-prone FMS together with the first strongly connected component (SCC) by using INA (Integrated Net Analyzer). The number of legal states computed from the first SCC that includes the initial marking represents the LZ of a RG. The proposed algorithm is implemented as an executable program. The number of legal states of a deadlock controller can easily and correctly be computed by using the proposed method and tool. Several well-known examples of FMS are considered to illustrate the applicability and the effectiveness of the proposed method.

Cite this article as: Gelen G, Uzam M. Computation of the number of legal states for petri net-based deadlock prevention problems. Sigma J Eng Nat Sci 2023;41(3):493–502.

***Corresponding author.**
*E-mail address: gokhan.gelen@btu.edu.tr
This paper was recommended for publication in revised form by Regional Editor İhsan Kaya

INTRODUCTION

A deadlock in modern manufacturing systems such as flexible manufacturing systems (FMS) or automated manufacturing systems (AMS), is an undesirable problem that should be absolutely resolved. Their existence frequently spoils the utilization of resources and may lead to devastating effects on the operation of such systems [1]. Petri nets (PN) are the most common algebraic and graphical tools for solving deadlock problems in FMS. Several analysis and control methods have been proposed for the solution of deadlock problems [2-25]. Analysis techniques related to deadlock problems are usually classified into two groups: structural analysis and reachability graph analysis [2]. The former is used to obtain deadlock prevention policies by using special PN objects such as siphons or resource-transition circuits [5, 14-16, 22-25]. The latter use the reachability graph (RG) that fully reflects the behavior of a system [4, 6-9, 13-15, 17-21]. While reachability graphs based methods provide optimal solutions, methods using structural analysis provide suboptimal solutions in most cases.

The RG of a Petri net suffering from deadlocks can be divided into two parts as a live-zone (LZ) and a deadlock zone (DZ) [4]. In view of control, we can classify the markings into four groups as deadlock, bad, dangerous and good markings. A deadlock marking represents a dead system state, from where the system does not change its current state. The initial marking cannot be reached from some markings that are called bad markings. A dangerous marking can reach a good, a bad or a deadlock marking depending on the control mechanism. The rest of markings except deadlock, bad and dangerous ones are classified as good markings. The zone in a reachability graph containing dead and bad markings is called DZ. The rest of the reachability graph is called the live-zone (LZ) that represents the maximally permissive behavior [9], from the viewpoint of deadlock i.e., legal markings are composed of good and dangerous markings. The number of legal markings (states) reachable under a deadlock prevention control policy has been regarded as a performance criterion. In order to assess the behavioral permissiveness of a control policy, it is important to know the number of legal states that must be provided by an optimal deadlock prevention supervisor.

Computation tools for Petri net models are utilized by researchers for different purposes such as analysis, code generation, simulation, animation and model checking. A database for PN tools and some comparisons can be found in [26, 27]. Although there are a lot of Petri Net tools currently available, INA (Integrated Net Analyzer) [3] is widely used in PN-based deadlock prevention problems [4-9, 21-25, 28, 29]. By using INA, structural properties, liveness analysis and reachability analysis of a PN can be obtained. By using INA, it is possible to compute the LZ as the first SCC for a Petri Net model of an FMS suffering from deadlock or livelock problem. Then the remaining SCCs constitute the DZ of the model, that may be containing not only bad and dead markings, but also livelocks. On the other hand further computations are necessary to find out the exact number of states in LZ and in DZ.

To date, for a given RG of a PN model of an FMS suffering from deadlocks, no study has been reported to provide the number of reachable legal states exist within the LZ of the given RG. In this paper, a method is proposed for the computation of the number of legal states that must be provided by an optimal deadlock prevention policy in the form of Petri net formalism. The proposed method makes use of the reachability graph (RG) analysis results of a given Petri net model of an FMS prone to deadlocks by using INA. The RG consists of LZ and DZ. If the LZ can be classified, the other one can also be classified as the rest of the RG. LZ consists of the first strongly connected component (SCC) of a RG that includes the initial marking. Therefore, in this paper, a method is proposed to compute the number of states of a given first SCC. The proposed method is implemented as an executable program. This software accepts RG and the first SCC as inputs and then computes the number of states in the LZ and computes the number of the states in DZ by subtracting the number of states in the LZ from the total number of all states in the related RG. In the literature some studies do not present the optimal live behavior of PN models correctly. By using the proposed algorithm and tool, the correct number of legal states in LZ and in DZ can be computed accurately. Thus, the method proposed in this paper provides an important contribution in order to determine the behavioral permissiveness of a Petri net based deadlock control policy. The applicability and the effectiveness of the proposed method is demonstrated by cosidering several well-known examples of FMS.

The paper is organized as follows. Section 2 recalls some notations of PNs and SCCs. Section 3 illustrates the deadlock analysis by means of a RG and SCCs. The proposed method and its algorithm are presented in Section 4. Some examples from the relevant literature are given in Section 5. Finally, conclusions are provided in Section 6.

PRELIMINARIES

Petri Nets

Petri nets are a commonly used formal tool for the study of deadlock problems in FMS. A Petri is a five-tuple $PN = (P, T, F, W, M_0)$, where P is the set of places and T is the set of transitions. P and T sets are finite, nonempty, and disjoint. $F \subseteq (P \times T) \cup (T \times P)$ is the flow relation of the PN. W: $(P \times T) \cup (T \times P) \rightarrow N = \{0,1,2,...\}$, and $W(x,y) = 0$ if $(x,y) \notin F$. W is the weight function of arcs. A marking in PN is a mapping $M:P \rightarrow N$. M_0 denotes the initial marking of PN [10, 12].

The preset of a node (transition or places) x is defined as $\bullet x = \{y \in P \cup T | (y; x) \in F\}$ and the postset of node (transition or places) x is defined as $x\bullet = \{y \in P \cup T | (x; y) \in F\}$. PNs can be graphically represented. Circles and squares (or bars) are used to represent places and transitions, respectively. Tokens are black dots that are put in places to indicate the marking (state)

of the system. In a PN, no arc may connect two places or two transitions. The weight of an arc is labeled by a number. An arc without a label, means that its weight is equal to one.

The count of tokens in p at M is presented by $M(p)$. *If* $M(p) > 0$, this shows that p is marked by M. The enabling of a transition t at marking M is specified by $M[t\rangle$, *if* $\forall p \in {\bullet}t$, $M(p) \geq W(p,t)$. While each input place $p \in {\bullet}t$ is marked with at least $w(p,t)$ tokens, transition t is said to be enabled. If a transition is enabled at marking M, it can be fired. The firing of a transition is denoted as $M[t\rangle M'$. The firing of a transition t at M can lead to reach M'. $R(N,M_0)$ indicates the set of all possible markings reachable from M_0. The reachability of M′ from M is denoted by $M[\sigma\rangle M'$, where $\sigma = (t1, t2, \ldots, tn)$ is the firing sequence. In case there is no enabled transition at a marking $M \in R$, it is said to be the net system (N, M_0) contains a deadlock. This marking is named as a dead marking. The occurrence of a deadlock in a real FMS can cause severe damage to the system or even stop the entire system from operation. Therefore, deadlock resolution is an important consideration in the design and control phase of FMS.

Reachability Graph and Strongly Connected Components

The reachability graph (RG) of a PN is a directed graph that covers all of reachable markings and transitions among these markings. The RG is defined as $RG = (N, L, E, n_0)$, where N is the set of nodes (vertices) and each node $n_k \in N$ represents a reachable marking. L is a set of labels, where each $l_i \in L$ corresponds to firable transitions in any reachable marking of PN. E is the set of edges (transitions) such that $E = \{(n_k, l_i, n_{k+1}) \in E \backslash n_k, n_{k+1} \in N, l_i \in L\}$ and for each $e \in E$ node n_k is the initial node, v_{k+1} is the terminal node and l_i is a label of edge related to PN transitions. $n_0 = M_0$ is the initial node. A PN model and its RG are shown in Figure 1.

A chain in a graph is a sequence of vertices from one vertex to another using the edges (or arcs). If every pair of vertices is covered by a chain, this graph is called as connected. A graph is strongly connected if, every node (vertex) is reachable from every other nodes. For example, a strongly connected graph and a connected graph is presented in Figure 2.a and 2.b, respectively. The connected graph in Figure 2.b is not strongly connected whereas there is no path from state 'b' to state 'h'. The strongly connected component (SCC) of a graph is a strongly connected subgraph that is maximal. The SCC of a graph cannot contain any extra transition or node from original graph that cancel its strongly connection [11-12].

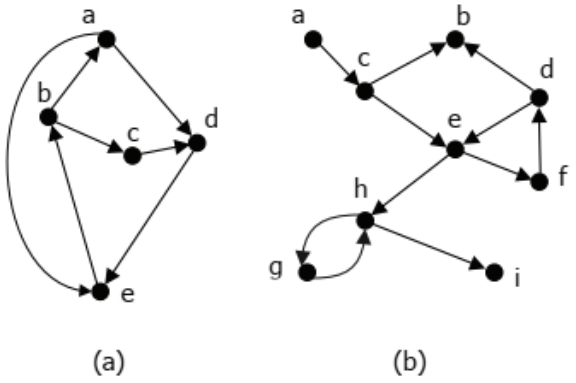

Figure 2. A strongly connected graph (a), A connected graph (b) [12].

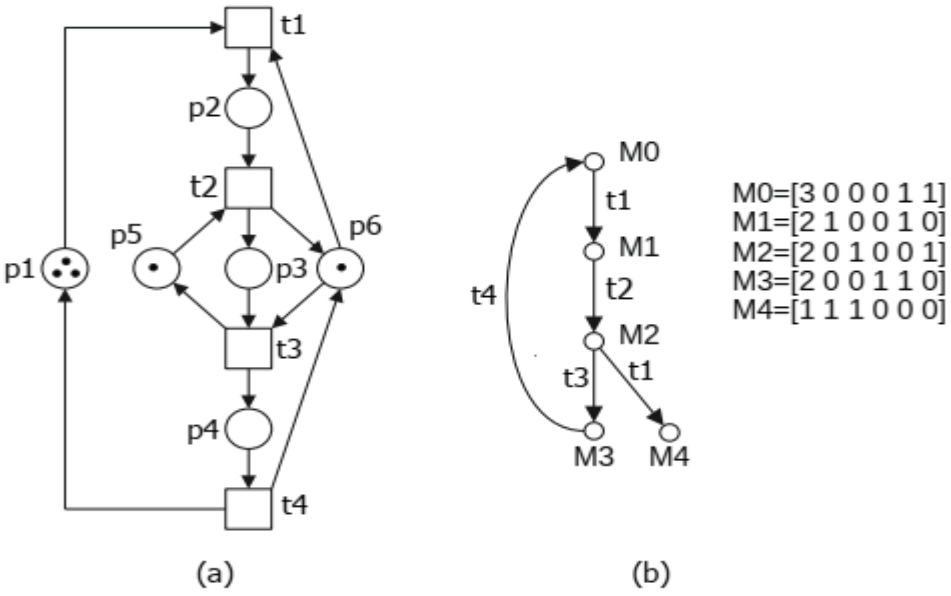

Figure 1. A PN model (a) and its reachability graph (RG) with markings (b).

Deadlock Analysis of PN Models by Means of Reachability Graphs

In this section, reachability graph-based deadlock analysis of PN models is briefly explained. The marking of a PN changes by the firing of enabled transitions. The RG of a PN model is constituted by its all possible markings and transitions between these markings. Indeed, a state in the RG indicates a marking of PN. States in a RG of PN can be divided into four groups such as deadlock states, bad states, dangerous states and good states. While dead states and bad states are included in DZ, the markings of RG other than DZ is defined as LZ which determines the optimal live behavior.

A small manufacturing system shown in Figure 3 is considered as an example. In this system, there are two machines namely M1 and M2 and a robot. Each machine has one part processing capacity at a time and the robot has only one part handling capacity. Input and output buffers are used for loading and unloading. It is considered that there are two different parts namely P1 and P2. Initially, it is assumed that there are no parts in the machines and robot. There are two-different production routes in this manufacturing system as follows:

PR1: Machine 1 → Robot → Machine 2

PR2: Machine 1 ← Robot ← Machine 2

This system can be modelled by using Petri net framework as shown in Figure 4. The model has 11 places and 8 transitions. The production routes PR1 and PR2 are modelled by using places p2, p5 and p8 (represents operation of Machine 1, Robot and Machine 2), and p10, p7 and p4 (represents operation of Machine 2, Robot and Machine 1), respectively. The number of tokens in p1 and p11 represents the count of concurrent activities for P1 and P2, respectively. The reachability graph of this system is shown in Figure 5 [13].

It can be seen from Figure 5 that there are 20 states in the reachability graph. The markings of PN represented as states of RG by using s1-s20 labels. The deadlock states are s8 and s13. When the system reaches these states, it is not possible to switch from these states to another state. If the system reaches to any one of bad states s10, s11 or s12, it can only reach other bad states or deadlock states. A system that has reached any deadlock or bad state will never be able to return to its initial state. Changing from dangerous states such as s2, s3, s4, s14, s15 and s16 to any one of a good, bad or deadlock state occurs according to the

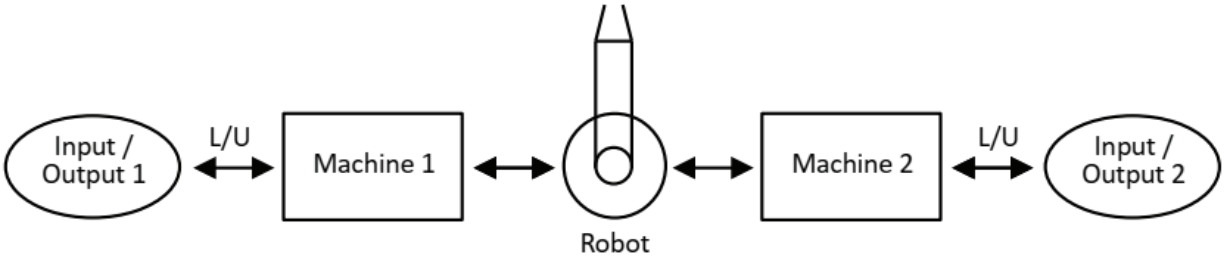

Figure 3. An example manufacturing system.

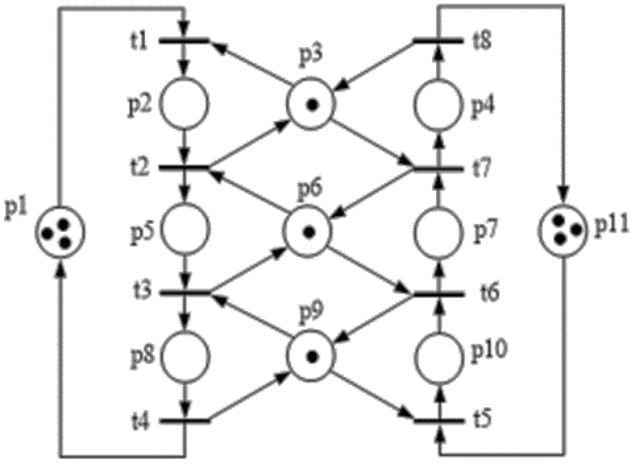

Figure 4. PN model of a system.

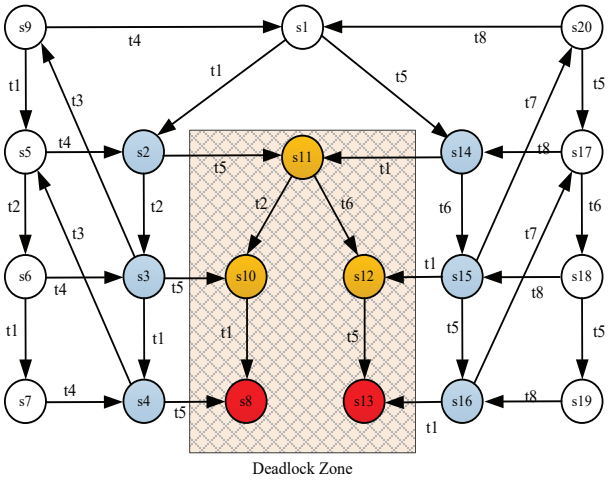

Figure 5. The reachability graph of example manufacturing system.

supervisory control. The dead zone for this RG is equal to the combination of bad and deadlock states. The number of states in the live zone can be easily computed by subtracting deadlock markings from all markings. In this RG, there are five states (markings) in the dead zone and there are 15 states in the LZ. The optimal behavior of this PN model is defined by the 15 markings in the LZ. INA is used to analyze PN model presented in Figure 4. The INA file representation of this PN model is depicted in Figure 6. Results for structural properties and reachability graph analysis are presented in Figure 7 and Figure 8, respectively. According to these results, the PN is not live and has two dead states as explained below. For such a small problem it is possible to compute the number of states both in LZ and DZ, but as the number of states becomes large it is necessary to do this computation by means of a computer program. In the following section, an algorithm to compute LZ and DZ is proposed.

```
p  m pre,post
1  3 4,1
2  0 1,2
3  1 2 8,1 7
4  0 7,8
5  0 2,3
6  1 3 7,2 6
7  0 6,7
8  0 3,4
9  1 4 6,3 5
10 0 5,6
11 3 8,5
@
```

Figure 6. INA file representation of the PN model.

COMPUTATION OF THE NUMBER OF LEGAL STATES

The SCCs of a PN model can be obtained by using INA. The first SCC contains the initial marking of PN model. The dead and bad markings are covered by all SCCs except for the first SCC of a given RG. Then, the LZ can be computed by counting the states in the first SCC. After obtaining the number of states in the LZ, the count of states in the DZ can be easily computed by subtracting the number of states of the LZ from that of the related RG. The proposed algorithm to compute the number of states both in LZ and in DZ is presented in the following. This algorithm is implemented in C programming language and named as "zone_analyzer.exe". After the computation, the results are both displayed

```
Information on elementary structural properties:

The net is not statically conflict-free.
The net is pure.
The net is ordinary.
The net is homogenous.
The net is not conservative.
The net is not subconservative.
The net is not a state machine.
The net is not free choice.
The net is not extended free choice.
The net is extended simple.
The net is not safe.
The net is not live and safe.
The net is marked.
The net is not marked with exactly one token.
The net is not a marked graph.
The net has a non-blocking multiplicity.
The net has no nonempty clean trap.
The net has no transitions without pre-place.
The net has no transitions without post-place.
The net has no places without pre-transition.
The net has no places without post-transition.
Maximal in/out-degree:  2
The net is connected.
The net is strongly connected.
```

Figure 7. Some structural properties of the PN model depicted in Figure 4.

```
Computation of the reachability graph

States generated: 20
Arcs generated: 34

Capacities needed:
Place  1  2  3  4  5  6  7  8  9 10 11
 Cap:  3  1  1  1  1  1  1  1  1  1  3
Dead states:
      8,  13,
Number of dead states found: 2
The net has dead reachable states.
The net is not live.
The deadlock-trap-property is not valid.
The net is not reversible (resetable).
The net has no dead transitions at the initial marking.
The net is not live, if dead transitions are ignored.
The net is bounded.
```

Figure 8. Reachability graph computation of the PN model depicted in Figure 4.

on the screen and reported in a text file by the software at the same time.

Algorithm: Computation of the number of states in LZ and DZ

Input: PN model as a .pnt file.

Definitions: *nbr_sts_lz*: number of states in the LZ, *nbr_sts_dz*: the number of states in the DZ, *t_nbr_sts*: total number of states in RG, *nbr_scc*: the number of states in the first SCC, *x*: the number of read data

Step 1: Compute the RG and then SCC of the given PN model by using INA.

// after the computation of the RG and SCCs, results are stored as "rg.sta" and "scc.res" files, respectively

Step 2: Compute the total number of states (*t_nbr_sts*) in RG from rg.sta file.

Step 3: Obtain the first SCC from "scc.res" file and store it in an array named as *Scc*[ ].

//*Scc* [ ], the first SCC covers the LZ. The data structure of *Scc*[ ] is in the form of [number ,.., number, number ,.., number, number, number ,..,number, …]. For example, a first SCC is 1 .. 7, 9, 14 ..20.

Step 4: Read a part of *Scc*[ ] to the first comma in the form of [*a, s, b*].

//*a* and *b* are integer numbers, *s* is a string.

Step 5: Find the number of read data as *x*.

Step 6: If *x*=3 then *nbr_scc*= *nbr_scc* + (*b*-*a*+1).

Else if *x*=1 then *nbr_scc* = *nbr_scc* + 1.

Step 7: If elements of *Scc*[ ] array are not finished then return Step 4.

Else continue

Step 8: *nbr_sts_dz* = *t_nbr_sts* - *nbr_scc*

nbr_sts_lz = *nbr_scc*

Output: The number of states in RG, the number of states in

DZ, the number of states in LZ.

End of Algorithm.

A short user procedure for zone_analyzer is presented as follows.

- Create a folder including INA (Integrated Net Analyzer) files.
- Copy the "Command.ina" file into the same directory.
- Define the Petri Net model of the system in the "input.pnt" INA file.
- Run "INAwin32.exe" and press "Y" to follow the predefined procedures in "Command.ina" file. After the computation of the Reachability Graph (RG) and Strongly Connected Components (SCCs), results are stored as "rg.sta" and "scc.res" files in same folder, respectively.
- Run "zone_analyzer.exe".
- Observe the number of states in Live Zone (LZ) and in Deadlock Zone (DZ) from the file "results.txt".

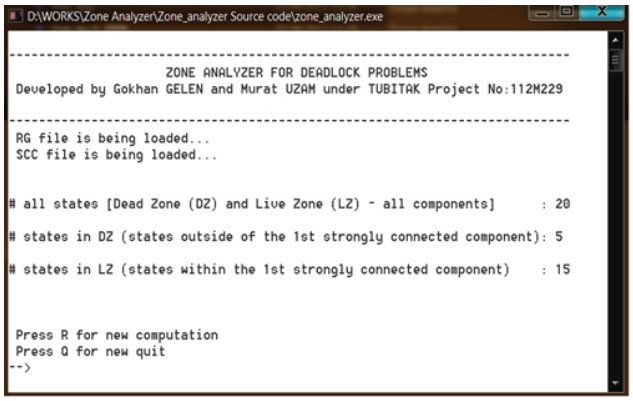

Figure 9. A screenshot of "zone_analyzer.exe" program.

Example Computations

This section reports the application of proposed technique to a few well-known FMS and AMS models from the literature. The proposed algorithm is implemented as an executable program called "zone_analyzer.exe". The developed program, files of presented examples and a short user guide can be downloaded from the following website: https://github.com/gokhangelen/zone-analyzer

Computations for an S³PR Model

As the first example model, the S³PR (Systems of simple sequential processes with resources) PN model is considered. As shown from Figure 10, the S³PR model has 15 places and 11 transitions. This model has been frequently considered in the literature [5, 14-16, 21, 22]. The RG analysis of this model was performed by using INA. The INA file representation of S³PR model is depicted in Figure 11. The results show that the net has 261 states and 933 arcs.

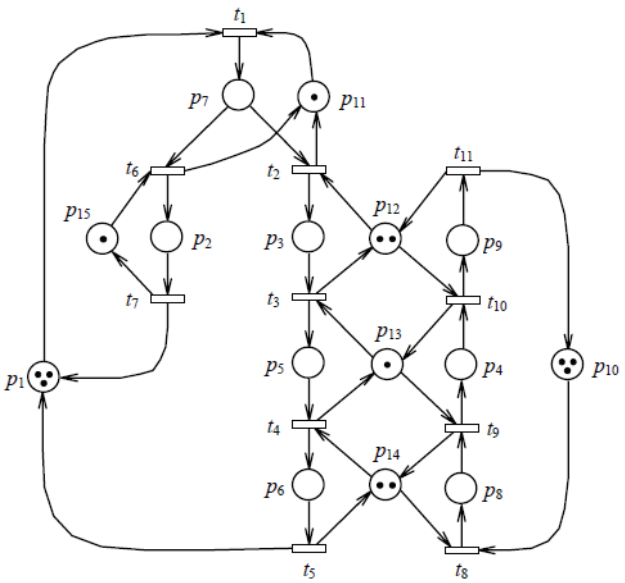

Figure 10. An S³PR model.

```
p  m pre,post
 1  3  7 5,1
 2  0  6,7
 3  0  2,3
 4  0  9,10
 5  0  3,4
 6  0  4,5
 7  0  1,6,2
 8  0  8,9
 9  0  10,11
10  3  11,8
11  1  6 2,1
12  2  3 11,2 10
13  1  4 10,3 9
14  2  5 9,4 8
15  1  7,6
@
```

Figure 11. INA file representation of the S³PR model depicted in Figure 10.

```
---------------------------------------------
ZONE ANALYZER FOR DEADLOCK PROBLEMS
Developed by Gokhan GELEN and Murat UZAM under
TUBITAK Project No:112M229
---------------------------------------------

# all states [Dead Zone (DZ) and Live Zone (LZ) - all components]   : 261

# states in DZ (states outside of the 1st strongly connected component) : 29

# states in LZ (states within the 1st strongly connected component)    : 232
```

Figure 12. Results for PN model of an S³PR.

The net is not live and has one deadlock state. The first SCC is computed as "1 .. 53, 65 .. 76, 92 .. 100, 104 .. 261," for this PN model. The count of states in the LZ can be easily determined by using the implemented "zone_analyzer.exe" program. According to the results obtained from proposed algorithm, as shown in Figure 12, the PN model has 261 markings and 29 of which are in the DZ. For this PN model, the optimal behavior is characterized by the LZ containing 232 good markings.

Computations for an S³PGR² Model

A C/D-RAS (conjunctive/disjunctive resource allocation Systems) [18] consisting of five robots, three loading buffers and three unloading buffers is used as the second example. The layout of this system is shown in Figure 13. The handling capacity of the first six robots is seven and handling capacity of the last robot is six. An S³PGR² (system of simple sequential processes with general resource requirement) model for this manufacturing cell is shown in Figure 14. S³PGR² models have been frequently studied in the literature [18-20]. S³PGR² model of this manufacturing cell has 26 places and 22 transitions. The RG and SCC analysis of this model is performed by using INA. The INA file representation of S³PGR² model is depicted in Figure 15. The LZ is computed by using the proposed "zone_analyzer.exe" program. The results for this PN model are shown in Figure 16. According to these results, the PN model has 334698 markings and 120 of which are in DZ. Thus, the optimal behavior is characterized by the LZ containing 334578 markings for this PN model. Note that in [18] the optimal live behavior of this PN model is not presented correctly. This result shows the significance of the method proposed in this paper.

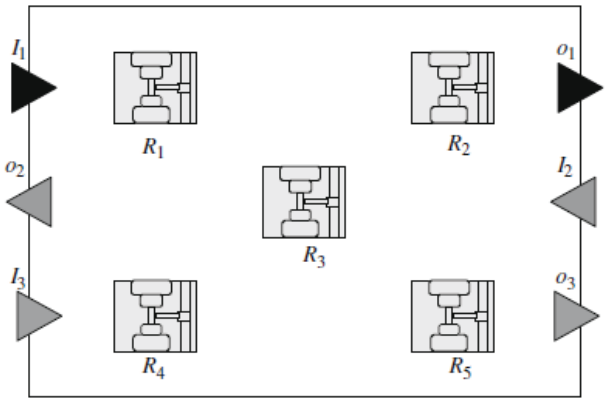

Figure 13. Layout of the manufacturing cell from [17].

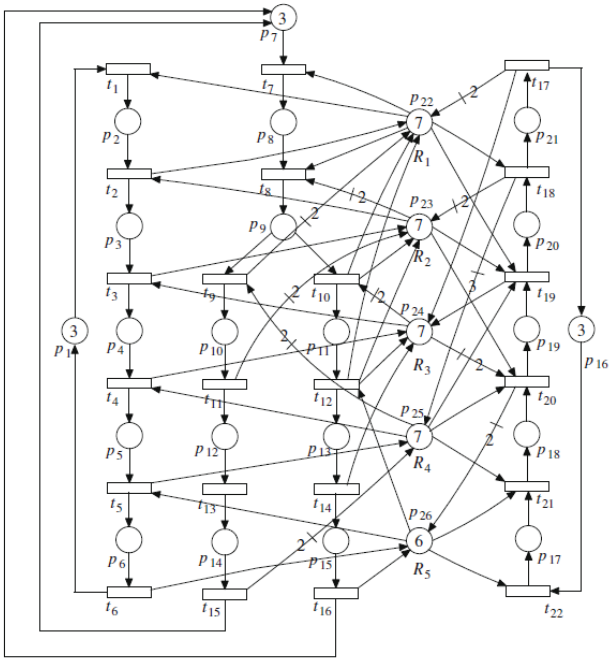

Figure 14. An S³PGR² model.

```
p  m pre,post
1  3 6,1
2  0 1,2
3  0 2,3
4  0 3,4
5  0 4,5
6  0 5,6
7  3 15 16,7
8  0 7,8
9  0 8,9 10
10 0 9,11
11 0 10,12
12 0 11,13
13 0 12,14
14 0 13,15
15 0 14,16
16 3 17,22
17 0 22,21
18 0 21,20
19 0 20,19
20 0 19,18
21 0 18,17
22 7 2 9:2 10 12 17:2,1 7 8 18 19
23 7 3 10 11:2 12 18:2,2 8:2 19 20
24 7 4 12 14 17 19,3 10:2 20:2
25 7 5 15:2 18:3,4 9:2 19 20 21
26 6 6 16 20:2,5 12 21 22
@
```

Figure 15. INA file representation of the S³PGR² model depicted in Figure 13.

```
---------------------------------------------------------------------------
        ZONE ANALYZER FOR DEADLOCK PROBLEMS
   Developed by Gokhan GELEN and Murat UZAM under
        TUBITAK Project No:112M229
---------------------------------------------------------------------------

# all states [Dead Zone (DZ) and Live Zone (LZ) - all components]     : 334698

# states in DZ (states outside of the 1st strongly connected component) : 120

# states in LZ (states within the 1st strongly connected component)    : 334578
```

Figure 16. Results for PN model of an S³PGR².

Computations for an S⁴PR Model

In this example, an automated manufacturing system (AMS) that produces three parts P1, P2, and P3, is considered. The layout of this AMS is shown in Figure 17. There are five resources named as R1, R2, R3, R4 and R5. Each of which has the capacity of eight. The PN model of this system is not given correctly in [21]. The corrected PN model for this AMS is illustrated in Figure 18. The RG and SCC analysis of this model is performed by using INA. The INA file representation of S⁴PR model is depicted in Figure 19. The LZ is computed by using the proposed "zone_analyzer.exe" program. The results for this PN model are shown in Figure 20. According to these results, the PN model has 421496 markings and 132 of which are in DZ. Thus, the optimal behavior is characterized by the LZ containing 421358 markings for this PN model.

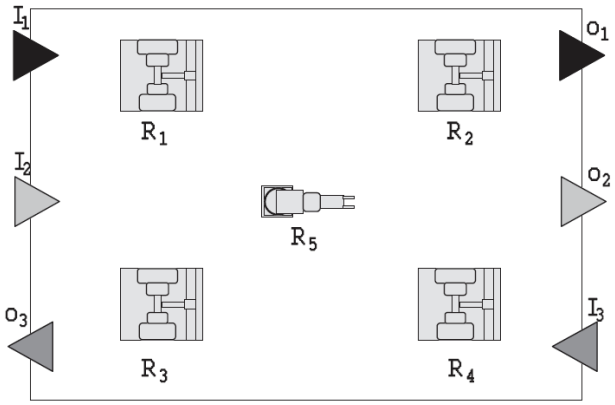

Figure 17. An AMS example from [20].

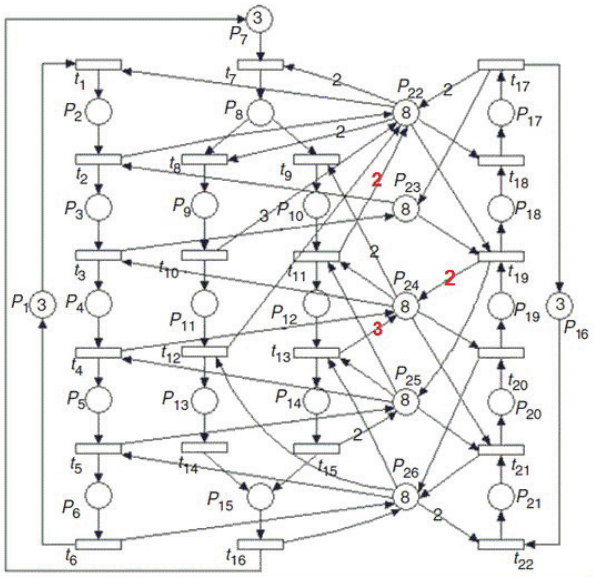

Figure 18. The (corrected) PN model of the AMS.

In [21], it is assumed that the optimal live behavior of the PN model shown in Figure 18 is the same as all the number of states exist, i.e., 421496 states, in the uncontrolled RG of the PN model. Therefore, the result provided here provides another mistake made in the literature when computing the optimal behavior of PN based liveness enforcing supervisors.

```
p  m pre, post
1  3  6,1
2  0  1,2
3  0  2,3
4  0  3,4
5  0  4,5
6  0  5,6
7  3  16,7
8  0  7,8 9
9  0  8,10
10 0  9,11
11 0  10,12
12 0  11,13
13 0  12,14
14 0  13,15
15 0  14 15,16
16 3  17,22
17 0  18,17
18 0  19,18
19 0  20,19
20 0  21,20
21 0  22,21
22 8  2 10:3 11:2 12 17:2,1 7:2 8:2 18 19
23 8  3 17,2 19
24 8  4 13:3 19:2,3 9:2 11 20 21
25 8  5 15:2 19,4 11 13 21
26 8  6 16 20 21,5 12 13 22:2
@
```

Figure 19. INA file representation of the PN model depicted in Figure 18.

```
---------------------------------------------------------------------------
        ZONE ANALYZER FOR DEADLOCK PROBLEMS
   Developed by Gokhan GELEN and Murat UZAM under
        TUBITAK Project No:112M229
---------------------------------------------------------------------------

# all states [Dead Zone (DZ) and Live Zone (LZ) - all components]     : 421496

# states in DZ (states outside of the 1st strongly connected component) : 138

# states in LZ (states within the 1st strongly connected component)    : 421358
```

Figure 20. Results for PN model shown in Figure 18.

CONCLUSION

In this paper, a solution for the computation of the number of states in the live zone (LZ) that describes the optimal (maximally permissive) behavior of a Petri net (PN) based deadlock prevention policy is presented. The proposed algorithm is implemented as an executable program. It accepts the reachability graph and the first strongly connected component (SCC) obtained from the most popular PN tool INA as input and computes the exact number of all legal states. The feasibility and applicability of the proposed technique is shown by using well-known Petri net models from the literature. In addition, this paper also reports that in some papers from the literature the number of legal states for S^3PGR^2 and S^4PR nets are computed incorrectly. This fact further shows the importance of the method reported in this paper. The proposed tool can be used for both structural analysis-based and reachability graph analysis-based liveness enforcing supervisors. Thanks to the developed program, the number of legal states that is considered as a quality measure of deadlock controller can be easily and accurately computed. The proposed method makes use of the RG analysis that usually requires a complete enumeration of reachable states. Therefore, it suffers from the state explosion problem.

ACKNOWLEDGEMENTS

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. The authors would like to thank the Editor and five anonymous referees whose comments and suggestions greatly helped us to improve the presentation and the quality of this paper.

AUTHORSHIP CONTRIBUTIONS

Authors equally contributed to this work.

DATA AVAILABILITY STATEMENT

The authors confirm that the data that supports the findings of this study are available within the article. Raw data that support the finding of this study are available from the corresponding author, upon reasonable request.

CONFLICT OF INTEREST

The author declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

ETHICS

There are no ethical issues with the publication of this manuscript.

REFERENCES

[1] Li ZW, Wu NQ, Zhou MC. Deadlock control of automated manufacturing systems based on petri nets-a literature review. IEEE Trans Syst Man Cybern C Appl Rev 2012;42:437–462. [CrossRef]

[2] Chen YF, Li ZW. Optimal supervisory control of automated manufacturing systems. 1st ed. UK: Taylor & Fracis Group; 2013. [CrossRef]
[3] Informatik. INA Integrated net analyzer a software tool for analysis of petri nets, Available at: http://www.informatik.hu-berlin.de/~starke/ina.html. Accessed on May 26, 2023.
[4] Uzam M. An optimal deadlock prevention policy for flexible manufacturing systems using Petri net models with resources and the theory of regions. Int J Adv Manuf Tech 2002;19:192–208. [CrossRef]
[5] Lin R, Yu Z, Shi X, Dong L, Nasr EA. On multi-step look-ahead deadlock prediction for automated manufacturing systems based on petri nets. IEEE Access 2020;8:170421–170432. [CrossRef]
[6] Shaoyong L, Chunrun Z. A deadlock control algorithm using control transitions for flexible manufacturing systems modelling with Petri nets. Int J Syst Sci 1994;51:771–785. [CrossRef]
[7] Uzam M, Zhou MC. An iterative synthesis approach to Petri net-based deadlock prevention policy for flexible manufacturing systems. IEEE Trans Syst Man Cybern Syst Hum 2007:37;362–371. [CrossRef]
[8] Uzam M, Li ZW, Zhou MC. Identification and elimination of redundant control places in petri net based liveness enforcing supervisors of FMS. Int J Adv Manuf Tech 2007;35:150–168. [CrossRef]
[9] Hu M, Yang S, Chen Y. Partial reachability graph analysis of petri nets for flexible manufacturing systems. IEEE Access 2020;8:227925–227935. [CrossRef]
[10] Li ZW, Zhou MC. Deadlock resolution in automated manufacturing systems: Novel petri net approach. 1st ed. Berlin: Springer; 2009.
[11] Cormen T, Leiseson C, Rivest R, Stein C. Introduction to Algorith. 2nd ed. Cambridge: MIT Press; 2001.
[12] David R, Alla H. Discrete Continious and Hybrid Petri Nets. 1st ed. Berlin: Springer; 2005.
[13] Gelen G, Uzam M, Li ZW. A new method for the redundancy analysis of Petri net-based liveness enforcing supervisors. Trans Inst Meas Control 2017;39:763–780. [CrossRef]
[14] Huang Y, Jeng M, Xie X, Chung S. Deadlock prevention policy based on Petri nets and siphons. Int J Prod Res 2001;39:283–305. [CrossRef]
[15] Li ZW, Zhou MC. Elementary siphons of petri nets and their application to deadlock prevention in flexible manufacturing systems. IEEE Trans Syst Man Cybern Syst Hum 2004;34:38–51. [CrossRef]
[16] Piroddi L, Cordone R, Fumagalli I. Selective siphon control for deadlock prevention in Petri nets. IEEE Trans Syst Man Cybern Syst Hum 2008;38:1337–1348. [CrossRef]
[17] Ezpeleta J, Colom JM, Martinez J. A petri net based deadlock prevention policy for flexible manufacturing systems. IEEE Trans Rob Autom 1995;11:173–184. [CrossRef]
[18] Hu H, Li ZW. Local and global deadlock prevention policies for resource allocation systems using partially generated reachability graphs. Comput Ind Eng 2009;57:1168–1181. [CrossRef]
[19] Chao DY, Chen JT, Yu F. A novel liveness condition for S^3PGR^2. Trans Inst Meas Control 2013;35:131–137. [CrossRef]
[20] Shih YY, Chao DY, Chiu CY. A New MIP Test for S^3PGR^2. In: Chou SY, Trappey A, Pokojski J, Smith S, editors. Global Perspective for Competitive Enterprise, Economy and Ecology: Proceedings of the 16th ISPE International Conference on Concurrent Engineering; 2009 July 20-24; Taipei, Taiwan: Springer; 2009. pp. 41–52.
[21] Hu HS. An iterative deadlock prevention approach for automated manufacturing systems. Trans Inst Meas Control 2011;33:59–76. [CrossRef]
[22] Guo X, Wang S, You D, Li ZW, Jiang X. A siphon-based deadlock prevention strategy for S3PR. IEEE Access 2019;7:86863–86873. [CrossRef]
[23] Zhuang Q, Dai W, Wang S, Ning F. Deadlock prevention policy for S4PR nets based on siphon. IEEE Access 2018;6:50648-50658. [CrossRef]
[24] Li ZW, Zhou MC, Wu NQ. A survey and comparison of petri net-based deadlock prevention policies for flexible manufacturing systems. IEEE Trans Syst Man Cybern C Appl Rev 2008;38:173–188. [CrossRef]
[25] Liu GY, Barkaoui K. A survey of siphons in Petri nets. Inf Sci 2016;363:198–220. [CrossRef]
[26] Thong WJ, Ameedeen MA. A survey of petri net tools. Lect Notes Electr Eng 2015;315:537–551. [CrossRef]
[27] Informatik. Petri nets tool database. Available at: https://www2.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html Accessed on May 26, 2023.
[28] Huang B, Zhou MC, Wang C, Abusorrah A, Al-Turki Y. Deadlock-free supervisor design for robotic manufacturing cells with uncontrollable and unobservable events. IEEE/CAA J Autom Sin 2021;8:597–605. [CrossRef]
[29] Lin R, Yu Z, Shi X, Dong L, Nasr EA. On multi-step look-ahead deadlock prediction for automated manufacturing systems based on petri nets. IEEE Access 2020;8:170421–170432. [CrossRef]