Contents

General Information

Petri Net Basis Analyzer (PNBA) is a software for the analysis of Petri net based on the Basis Marking Approach. It is developed using Python and works on Microsoft Windows operating system. **PNBA 0.1** is developed at School of Electro Mechanical Engineering, Xidian University (Prof. Ziyue Ma and his colleague Minqiang Zou) with joint effort from Southwest Jiaotong University (Prof. Yin Tong).

The first "Basis Reachability Graph" (BRG) for the purpose of fault diagnosis in partially observed Petri nets was reported in [1]. Some preliminary work using *minimal explanations* technique can be dated back even earlier [2][3]. Later in 2017 the work [4] generalized this technique for marking space abstraction in arbitrary Petri nets, not only for partially observed nets. The main advantage of this technique is that only part of the reachability space of a net, the so-called *basis markings*, is enumerated. All other markings in the reachability set can be characterized by a linear algebraic system. Hence, using basis marking analysis can reduce the computational load and greatly alleviate the state explosion issue than exploring the reachability space by brute-force. So far, the basis marking approach has been effectively used in the field of fault diagnosis/diagnosability [5], controllability [6], and marking estimation and opacity [7] in Petri nets.

Note that PNBA is not intended to be developed as a generic design tool: it is simply a tool for teaching and research. As such, it only works with mediate size of nets (say, the number of places and transitions <99). On the other hand, PNBA is *portable* and *instantly usable* (i.e., you do not need to set environments by installing compilers and so on), and the input/output of PNBA are *human-friendly*, easy for users to read and comprehend (which sacrifices some computational efficiency). Hence, PNBA is particularly suitable to design the examples for your academic papers and/or the illustrations for fresh master/PhD students who wish to use and/or extend this technique in their work.

The software has four functional modules, namely:

- *TE Generator*: to automatically compute a set of explicit transitions;
- *BRG Generator*: to compute a BRG of a net;
- *Sequence Designer*: to compute a sequence of transitions that drives a net from a given source marking to a set of target markings;
- *Marking Estimator*: to compute the BRG observer from a BRG, and to compute the set of consistent basis markings with respect to a given observation.

Basics for Petri Nets and Basis Markings

We assume that the users are familiar with the basic notions of Petri nets. For technical details of basis markings and basis reachability graphs, please refer to [4].

Petri Net

A Petri net is a four-tuple $N = (P, T, Pre, Post)$, where P is a set of m *places*, T is a set of n *transitions*, $Pre : P \times T \to \mathbb{N}$ and $Post : P \times T \to \mathbb{N}$ are the *pre-* and *post-incidence matrices* in $\mathbb{N}^{m \times n}$. A marked net $\langle N, M_0 \rangle$ is a net N with an initial marking M_0, an m-component vector that assigns each place a non-negative integer number of tokens.

Minimal Explanations

Given a Petri net $N = (P, T, Pre, Post)$, a pair $\pi = (T_E, T_I)$ is called a *basis partition* of T if (1) $T_I \subseteq T$, $T_E = T \setminus T_I$; and (2) the T_I-induced subnet is acyclic. The sets T_E and T_I are called the set of *explicit transitions* and the set of *implicit transitions*, respectively.

Given a Petri net $N = (P, T, Pre, Post)$, a basis partition $\pi = (T_E, T_I)$, a marking M, and a transition $t \in T_E$, we define:

- $\Sigma(M, t) = \{\sigma \in T_I^* \mid M[\sigma\rangle M', M' \geq Pre(\cdot, t)\}$ as the set of *explanations* of transition t at marking M;
- $Y(M, t) = \{\mathbf{y}_\sigma \in \mathbb{N}^{|T|} \mid \sigma \in \Sigma(M, t)\}$ as the set of *explanation* vectors of transition t at marking M;
- $Y_{min}(M, t)$ denotes the set of all minimal elements of $Y(M, t)$, i.e., the *minimal explanation vectors*.

Basis Marking and Reachability Graph

The set of its *basis markings* $\mathcal{M}_{basis}$ is defined as:

- $M_0 \in \mathcal{M}_{basis}$;
- If $M \in \mathcal{M}_{basis}$, then for all $t \in T_E$, for all $\mathbf{y} \in Y_{min}(M, t)$, $M' = M + C \cdot \mathbf{y} + C(\cdot, t) \Rightarrow M' \in \mathcal{M}_{basis}$.

The *basis reachability graph* (BRG) is a finite state automaton $\mathcal{B} = (\mathcal{M}_{basis}, Tr, \Delta, M_0)$, where

- the state set $\mathcal{M}_{basis}$ is the set of *basis markings*;
- the event set Tr is the set of pairs $(t, \mathbf{y}) \in T_E \times \mathbb{N}^{|T|}$;
- the transition relation $\Delta \subseteq \mathcal{M}_{basis} \times Tr \times \mathcal{M}_{basis}$ is:

$$\Delta = \{(M_1, (t, \mathbf{y}), M_2) \mid \mathbf{y} \in Y_{min}(M_1, t), t \in T_E, M_2 = M_1 + C \cdot \mathbf{y} + C(\cdot, t)\}$$

- the initial state is the initial marking $M_0 \in \mathcal{M}_{basis}$.

> Proposition [4]: Given a marked net $\langle N, M_0 \rangle$ with $N = (P, T, Pre, Post)$ and its BRG $\mathcal{B} = (\mathcal{M}_{basis}, Tr, \Delta, M_0)$ with respect to $\pi = (T_E, T_I)$, the following two statements are equivalent:
>
> 1. there exist a marking M and a basis sequence $\sigma = \sigma_1 t_{i_1} \cdots \sigma_n t_{i_n} \sigma_{n+1}$ such that $M_0[\sigma\rangle M$;
> 2. there is a path in the BRG $\mathcal{B}$ $M_0 \xrightarrow{(t_{i_1}, \mathbf{y}_1)} M_{b,1} \xrightarrow{(t_{i_2}, \mathbf{y}_2)} \cdots \xrightarrow{(t_{i_n}, \mathbf{y}_n)} M_{b,n}$ such that $M \in \{M' \mid M_{b,n}[\sigma\rangle M', \sigma \in T_I^*\}$.

A basis partition for any net $\langle N, M_0 \rangle$ always exists (e.g., $T_E = T, T_I = \emptyset$ is always a valid basis partition) and is in general not unique, which leads to different basis marking spaces.

PNBA Quick Guide

PNBA has four modules, namely: *TE Generator*, *BRG Generator*, *Sequence Designer*, and *Marking Estimator*. Detailed guidelines for each module are described as follows.

Module 1: TE Generator

The function of this module is to automatically compute an explicit transition set T_E, or enlarge a given transition set to be a valid explicit transition set.

Input

Mandatory Input:

- A Petri net $\langle N, M_0 \rangle$ (see Input Petri Net Format below)

Optional Input:

- An initial set of explicit transitions T_{E0} (see Input TE Format below)

After selecting the input file, you can click the "Start" button to proceed. If the explicit transition file is not designated, the program will directly return a valid set of explicit transitions. Otherwise, the program will automatically expand the designated set (if any) to a *minimal* explicit transition set T_E. The term *minimal* is in the sense of set containment, not the set cardinality.

Output

The output of this module is a set of valid explicit transition set T_E. It is a **.txt** file with a single line consists of the names of explicit transitions, separated by comma. If initial set of explicit transitions T_{E0} is given, T_E will be a superset (not necessarily proper) of T_{E0}, i.e., $T_E \supseteq T_{E0}$.

Input Petri Net Format

PNBA accepts two types of input for a net N with an initial marking M_0. The designated source file will be referred to as the **Net File**. The number of places/transitions in the net are limited to 99.

From .ndr file (data file created by TINA toolbox)

PNBA can take the Petri net $\langle N, M_0 \rangle$ from a designated data file of **TINA** toolbox. **TINA** (http://www.laas.fr/tina/) is a powerful tool for the editing and analysis of Petri Nets. You may draw a Petri net N in **TINA** with initial marking M_0, save it, and then select the saved **.ndr** file as the input in PNBA. PNBA will automatically read the net and the initial marking in the **.ndr** file. Note that when creating a net in **TINA** we highly suggest you *not* to add temporal information (i.e., time) on transitions to your net.

From .txt file (plain text file)

PNBA can take the net $\langle N, M_0 \rangle$ from a designated **.txt** file that is edible by any plain text processor such as Microsoft Notepad. The text **Net File** must be in the following format.

- The first line contains two integers m and n that denote the number of places and transitions in your net, respectively.
- The next line is a comment line that contains a word "Pre".
- Then, in the next m lines, each line contains n integers (separated by comma). These lines denote the Pre matrix.
- The next line is a comment line that contains a word "Post".
- Then, in the next m lines, each line contains n integers (separated by comma). These lines denote the $Post$ matrix.
- The next line is a comment line that contains a word "M0".
- Then, the next line contains m integers (separated by comma) which denotes the initial marking M_0.

The three comment lines are just for the users' convenience to identify their input content: PNBA does not validate the content of the comment lines (so, after declaring m and n, you must sequentially place Pre, $Post$, and M_0 as described above). **Note: although the content of the comment lines are not validated, the lines must not be omitted**, otherwise the input validation will fail.

An example of the net file is the following.

```
4,6
Pre
2,3,4,1,5,0
0,5,4,3,1,4
3,1,3,5,3,2
4,3,1,2,2,4
Post
0,4,3,1,0,3
4,1,4,4,1,2
3,5,3,3,3,0
5,4,4,1,5,0
M0
4,1,5,1
```

Input TE Format

The set of initially given explicit transitions is input from a plain text file (.txt), called **TE0 File**. The **TE0 File** contains a single line that consists of the names of explicit transitions, separated by comma.

- If the net is inputted from an **.ndr** file, all names of explicit transitions in the **TE0 File** must occur in the net (case-sensitive).
- If the net is inputted from a **.txt** file, according to the columns of the Pre and $Post$ matrices, the name of the transitions in the net are automatically named "**t00**", "**t01**", "**t02**", and so on until "**t99**". As a result, the name of explicit transitions in the corresponding **TE0 File** should be something like "**t02, t05, . . .**" (letter 't' is case-sensitive).

An example of the **TE0 file** is the following.

```
t00, t01, t03
```

If the net is inputted from a **.ndr** file created by TINA and contains four transitions labeled by: *valve_on*, *valve_off*, *structed*, *unstructed*, a corresponding valid **TE0 file** may look like the following.

```
valve_on, valve_off, unstructed
```

Module 2: BRG Generator

The function of this module is to generate a Basis Reachability Graph of a Petri net with respect to a designated set of explicit transitions.

Input

Mandatory Input:

- A Petri net $\langle N, M_0 \rangle$ (same as Module 1, see Input Petri Net Format)
- A set of explicit transitions T_E

Input TE Format

The set of explicit transitions is input from a designated plain text file (.txt), called **TE File**. The **TE File** contains a single line that consists of the names of explicit transitions, separated by comma (same as the **TE0 file** in Module 1, see Input TE Format).

- If the net is inputted from an **.ndr** file, all names of explicit transitions in the **TE File** must occur in the net (case-sensitive).
- If the net is inputted from a **.txt** file, according to the columns of the Pre and $Post$ matrices, the name of the transitions in the net are automatically named "**t00**", "**t01**", "**t02**", and so on until "**t99**". As a result, the name of explicit transitions in the corresponding **TE File** should be something like "**t02, t05, . . .**" (letter 't' case-sensitive).
- Note: the output **TE file** of Module 1 can be directly used as the input of this module.

After selecting the input file, you can click the "Start" button to proceed.

Execution

When the "Start" button is clicked, PNBA will first read the files and validate the input data. PNBA can recognize several common types of invalid/typo-ed inputs, e.g., the initial marking contains negative tokens. In particular, PNBA can detect if the residue subnet of T_E (i.e., the *implicit subnet*) contains cycles and, if so, throw an error.

If everything is fine, PNBA will start computing the basis reachability graph of the net $\langle N, M_0 \rangle$ with T_E. The time it takes may vary which depends on (i) the size and the initial marking of the net, and (ii) the size of the corresponding basis reachability graph. During the computation, PNBA will continuously report in the log window the number of basis markings it has computed so far (for every 10,000 new basis markings).

> Empirically, for a moderate size of net (8 places, 9 transitions) with 10^6 basis markings, the computation is done in about 30 seconds on a desktop computer equipped with i5-3470 (an Intel CPU launched in 2012) on Windows 10 platform. The peak memory usage in this run is 250 megabytes.

It is worth noting that **PNBA does NOT detect the boundedness of the net nor of the basis reachability graph.** It is known [1] that the basis reachability graphs of some (but not all) unbounded nets are infinite. In such a case, PNBA will not terminate (you need to kill PNBA using Windows Task Manager). Since PNBA continuously reports the number of basis markings it has computed, if you observe in the log window that the number of basis markings has far exceeded your expectation and is still counting, then there may be such an unbounded issue with your net.

Output

The output of this function has three optional modes:

- **.txt(concise)** : results are written as a **.txt** file. The contents of the generated file include the basic information of the input net and its set of Basis Markings with respect to the designated set of explicit transitions. This mode is recommended when the whole Basis Reachability Graph is too large.
- **.txt(normal)** : results are written as a **.txt** file. The contents of the generated file include the basic information of the input net and its BRG. Note: such file may be very huge when the BRG is large, since both basis markings and the arcs will be outputted.
- **.brg** : the entire BRG data is written as a single **.brg** file which will be used for other module(s) of PNBA.

Module 3: Sequence Designer

The function of this module is to compute an optimal control sequence which drives a plant net from a source marking to a set of target markings without passing any forbidden markings. Details of this can be found in [8].

> Problem Description: Given a net $\langle N, M_0 \rangle$, a cost vector $\mathbf{c} = [c(t_1), \ldots, c(t_n)]^T$ that assigns each transition a cost to fire, a source marking $M_s \in R(N, M_0)$, a set of target markings $\mathcal{M}_{target}$, a set of forbidden markings $\mathcal{M}_{forbid}$, determine a control sequence σ_c such that:
>
> - by firing σ_c the net moves from M_s to some marking in $\mathcal{M}_{target}$ without passing forbidden markings in $\mathcal{M}_{forbid}$
> - the cost of firing σ_c is minimal.

Both the set of target markings $\mathcal{M}_{target}$ and the set of forbidden markings $\mathcal{M}_{forbid}$ can be defined by *generalized mutual exclusion constraints* (GMECs) [9], denoted by $(\boldsymbol{w}, k)$ and $(\boldsymbol{l}, b)$, respectively, i.e.:

$$\mathcal{M}_{target} = \mathcal{L}_{(\boldsymbol{w},k)} = \{M \in \mathbb{N}^{|P|} | \boldsymbol{w}^T \cdot M \le k\}$$
$$\mathcal{M}_{forbid} = \mathcal{L}_{(\boldsymbol{l},b)} = \{M \in \mathbb{N}^{|P|} | \boldsymbol{l}^T \cdot M \le b\}$$

On the other hand, $\mathcal{M}_{target}$ and $\mathcal{M}_{forbid}$ can also be defined by multiple number of GMECs.

- the target set $\mathcal{M}_{target}$ is defined by the **conjunction** of r_t GMECs $(\boldsymbol{w}_1, k_1), \dots, (\boldsymbol{w}_{r_t}, k_{r_t})$;
- the forbidden set $\mathcal{M}_{forbid}$ is defined by the **disjunction** of r_f GMECs $(\boldsymbol{l}_1, b_1), \dots, (\boldsymbol{l}_{r_f}, b_{r_f})$;

Precisely speaking:

$$\mathcal{M}_{target} = \bigcap_{i=1}^{r_t} \mathcal{L}_{(\boldsymbol{w}_i,k_i)} = \{M \in \mathbb{N}^{|P|} | (\forall i \in [1, r_t]) \ \boldsymbol{w}_i^T \cdot M \le k_i\}$$
$$\mathcal{M}_{forbid} = \bigcup_{i=1}^{r_f} \mathcal{L}_{(\boldsymbol{l}_i,b_i)} = \{M \in \mathbb{N}^{|P|} | (\forall i \in [1, r_f]) \ \boldsymbol{l}_i^T \cdot M \le b_i\}$$

Input

Mandatory Input:

- A Petri net $\langle N, M_0 \rangle$ (same as Modules 1, 2, see Input Petri Net Format)
- Metadata of the problem to be solved (see below)

Optional Input:

- A set of explicit transitions T_E (same as Modules 1, 2, see Input TE Format)

Metadata Input Format

PNBA takes the Metadata of a Problem from a designated plain text file **.txt**. in the following format. Here m and n below denote the number of places and transitions in the net, respectively.

- The first line is the designated cost vector $\boldsymbol{c}$ which contains n nonnegative real numbers (separated by comma).
- The second line contains one integer i denotes the number of $(\boldsymbol{w}_i, k_i)$ pairs.
- Then, in the next i lines, each line contains a $(\boldsymbol{w}_i, k_i)$ pair, where $\boldsymbol{w}_i$ is an m-dimensional integer vector (separated by comma) and k_i is an integer scalar. $\boldsymbol{w}_i$ and k_i are separated by comma.
- The next line contains one integer j denotes the number of $(\boldsymbol{l}_j, b_j)$ pairs.
- Then, in the next j lines, each line contains a $(\boldsymbol{l}_j, b_j)$ pair, where $\boldsymbol{l}_j$ is an m-dimensional integer vector(separated by comma) and b_j is an integer scalar. $\boldsymbol{l}_j$ and b_j are separated by comma.
- Note: vectors $\boldsymbol{c}, \boldsymbol{w}_i, \boldsymbol{l}_j$ must be enclosed in square brackets.

Here is an example of a metadata file. Given a net with $m = 6$ places $(p_{00}, p_{01}, p_{02}, p_{03}, p_{04}, p_{05})$ and $n = 4$ transitions $(t_{00}, t_{01}, t_{02}, t_{03})$, let the cost vector $\boldsymbol{c} = [1, 2, 3, 4]$. The target set $\mathcal{M}_{target}$ is defined by two GMECs: $[M(p_{00}) + 2M(p_{01}) \le 2] \wedge [M(p_{03}) \ge 3]$. The forbidden set $\mathcal{M}_{forbid}$ is defined by one GMEC: $M(p_{04}) + 2M(p_{05}) \le 2$. The corresponding metadata file is the following.

```
[1, 2, 3, 4]
2
[1, 2, 0, 0, 0, 0], 2
[0, 0, -1, 0, 0, 0], -3
1
[0, 0, 0, 0, 1, 2], 2
```

Execution

IMPORTANT: if the TE set you input is not valid, this module will automatically call Module 1 (with a line of log) to determine a valid set T_E before computing a control sequence σ_c. The set T_E eventually used for computing σ_c will be logged in the output file: please be sure it is what you want.

Output

The output of this module has two optional modes, both modes write the results to a **.txt** file that includes the basic information of the input and the generated optimal control sequence. The difference lies in the format of the output Optimal Control Sequence: in the *.txt (concise)* mode the firing of implicit transitions are depicted in vector form, while in the *.txt (normal)* mode their names are listed.

Module 4: Marking Estimator

This module has two functions, one (4a) is to generate the BRG Observer, the other (4b) is to generate the corresponding consistent basis markings with respect to a designated observation. For marking estimation in partially observed Petri nets, BRG observer, and consistent basis markings, please refer to [7].

Module 4a: Generate BRG Observer

Module 4 does not compute BRG observer from a net. You need to use Module 2 to first generate a **.brg** file for your net and input the BRG here. However, please be aware the way to input *label* in this module (see below).

Input:

Mandatory Input:

- A **.brg** file (generated by Module 2)

Optional Input:

- A **Label File** declaring the label of transitions in the net (see below)
  *Note: if the **.brg** file generated in Module 2 is from a net inputted from a **.txt** file, the **Label File** will become a **Mandatory Input**.*

Input the Label of transitions

If the **.brg** file generated in Module 2 is from a net inputted from an **.ndr** file, the labels in the **.ndr** file have already been encoded into the **.brg** file, although these labels are not shown in the output of Module 2. In such a case, you do not need to input a **Label File** (see below) here. On the other hand, if the **.brg** file generated in Module 2 is from a net inputted from a **.txt** file, then the **Label File** is **Mandatory**.

- **IMPORTANT: The Label File has priority.** If the **.brg** file is from an **.ndr** file and a **Label File** is inputted here, **PNBA will automatically use the labels in the Label File to proceed** (the label information in the **.brg** file is ignored), with a warning in the log window. This feature allows you to adjust the labels without revising the net in TINA, but be sure that the inputted labeling function is indeed what you want.

PNBA reserves term **'eps'** (note: case sensitive) as the symbol of unobservable label. Besides, when drawing the net in **TINA**, you only need to manually label observable transitions: when computing the BRG and generating the **.brg** file, Module 2 will automatically assign **'eps'** to all transitions that do not have a label.

Label File Format

The **Label File** is a plain text file (.*txt*). Each line of it consists of the name of a transition and its corresponding label, separated by comma. An example line is: "t01, a".

- The names of all transitions in the **Label File** must occur in the net (case-sensitive).
- Some combinations of special characters are reserved for the intermediate computation. In short, we recommend that each label only contains English letters, numbers, and **'-'**, **'_'**.
- **IMPORTANT:** If a transition is labeled multiple times in the **Label File**, PNBA will take the last label **WITHOUT a warning**. So, please examine the Label File carefully.

A **Label File** may look like the following.

```
t00, A
t01, B
valve_on, C
valve_off, C
```

Output

The output of this function has three optional modes:

- **console**: results are shown in the log window;
- **.txt**: the generated BRG observer are saved in a **.txt** file.
- **.obs**: the BRG observer is saved in an **.obs** file which is then used for module 4b as an input.

Module 4b: Generate Consistent Basis Markings

Module 4b takes the output from Module 4a to compute the set of *consistent basis markings* of a given word w.

Input

Mandatory Input:

- An **.obs** file (generated by Module 4a)
- An **.txt** file declaring an *observed word* w

Observed Word File Format

The file contains a single line that consists of the names of the observed labels (in order of occurrence), separated by comma. For example, the following file inputs word $w = abccd$:

```
a, b, c, c, d
```

Note that PNBA allows label with multiple letters. For example, the following file inputs the word that consists five consecutive events: *valve_on*, *valve_off*, *alarm*, *alarm*, *valve_on*:

```
valve_on, valve_off, alarm, alarm, valve_on
```

Output

The output of this function has three optional modes:

- **console (normal)**: output the consistent basis markings for each event in the word w in the log window.
- **console (concise)**: output the consistent basis markings in the log window.
- **.txt**: write all results in a **.txt** file.

Contact Us

If you find any problems/bugs when using PNBA, please contact zou_minqiang@163.com (Minqiang Zou); maziyue@xidian.edu.cn (Ziyue Ma); yintong@swjtu.edu.cn (Yin Tong).

Reference

[1] M. Cabasino, A. Giua, and C. Seatzu. Fault detection for discrete event systems using Petri nets with unobservable transitions. *Automatica*, 46(9):1531–1539, 2010.

[2] A. Giua and C. Seatzu. Fault detection for discrete event systems using Petri nets with unobservable transitions. In *Proceedings of Joint 44th Conference on Decision and Control and 2005 European Control Conference*, pages 6323–6328, Seville, Spain, 2005.

[3] G. Jiroveanu, R. Boel, and B. Bordbar. On-line monitoring of large Petri net models under partial observation. *Discrete Event Dynamic Systems*, 18(3):323–354, 2008.

[4] Z. Ma, Y. Tong, Z. Li, and A. Giua. Basis marking representation of Petri net reachability spaces and its application to the reachability problem. *IEEE Transactions on Automatic Control*, 62(3):1078–1093, 2017.

[5] N. Ran, H. Su, A. Giua, and C. Seatzu. Codiagnosability analysis of bounded Petri nets. *IEEE Transactions on Automatic Control*, 63(8):1192–1199, 2018.

[6] Z. Ma, Z. Li, and A. Giua. A method to verify the controllability of language specifications in Petri nets based on basis marking analysis. In *Proceedings of the 54th IEEE Conference on Decision and Control*, pages 1675–1681, Osaka, Japan, 2015.

[7] Y. Tong, Z. W. Li, C. Seatzu, and A. Giua. Verification of state-based opacity using Petri nets. *IEEE Transactions on Automatic Control*, 62(6):2823–2837, 2017.

[8] Z. Ma, M. Zou, J. Zhang, and Z. Li. Design of Optimal Control Sequences in Petri Nets Using Basis Marking Analysis. *IEEE Transactions on Automatic Control*, 2022, Early Access. DOI: 10.1109/TAC.2021.3106883.

[9] A. Giua, F. DiCesare, and M. Silva. Generalized mutual exclusion constraints for Petri nets with uncontrollable transitions. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics,* pages 947-949, Chicago, USA, 1992.