

State Space Search for Safe Time Petri Nets Based on Binary Decision Diagrams Tools: Application to Air Traffic Flow Management Problem

Mohamed Ali Kammoun, Nidhal Rezg, Zied Achour, Sadok Rezig

▶ **To cite this version:**

Mohamed Ali Kammoun, Nidhal Rezg, Zied Achour, Sadok Rezig. State Space Search for Safe Time Petri Nets Based on Binary Decision Diagrams Tools: Application to Air Traffic Flow Management Problem. Studies in Informatics and Control, 2016, 25 (1), pp.39-50. 10.24846/v25i1y201605 . hal-02968005

HAL Id: hal-02968005

https://hal.univ-lorraine.fr/hal-02968005

Submitted on 15 Oct 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

State Space Search for Safe Time Petri Nets Based on Binary Decision Diagrams Tools: Application to Air Traffic Flow Management Problem

Mohamed Ali KAMMOUN[1], Nidhal REZG[1,2], Zied ACHOUR[1,2], Sadok REZIG[1,2]

[1] Industrial Engineering and Production Laboratory of Maintenance,
Enim, Lorraine University, Metz, France.
mohamed-ali.kammoun@univ-lorraine.fr, nidhal.rezg@univ-lorraine.fr,
zied.achour@univ-lorraine.fr, sadok.rezig@univ-lorraine.fr

[2] ICN Business School,
Metz-Nancy, France.

Abstract: The highly concurrent time discrete event systems modeled by Time Petri Net (TPN) suffer from the problem of the state space explosion owing to a large number of accessible markings. To handle this problem, this paper proposes a new solution based on modelling in discrete time the reachable markings of TPN using a new structure so-called Time Reduced Ordered Binary Decision Diagrams (TROBDDs). In this work a new efficient methodology is presented to generate and store a big state space to deal with the time execution and memory space constraints. This new approach is used to resolve the Flight Rescheduling Problem (FRP) subject to capacity constraints due to adverse weather conditions. An optimization algorithm is proposed to minimize the cost function and determine the optimal flight plan according to the new capacity constraints. A number of instances on the FRP is presented in order to illustrate such approach, which allows us to save the memory space and CPU requirements.

Keywords: Discrete event system, Time Petri Net, binary decision diagram, rescheduling problem.

1. Introduction

The Time Petri Net (TPN) is obtained from the Petri Net by associating a time interval for each transition. The TPN is considered as a perfect tool to modelize a large class of time discrete models. Therefore, several existing methods for the TPN analysis used the enumeration of state space. Besides, for highly competitive systems, the size of state space grows exponentially.

To overcome this problem, several methods are proposed such as partial order unfolding [5] which are limited to safe TPN, and consist in transforming a TPN to an acyclic Petri net, by respecting on one hand the firing time constraints and on the other hand the partial order of the initial model. Our previous approach [7] is ensured by a new structure so-called Discrete Time Reachability Graph (DT-RG). Each node of the DT-RG, called macro-state, represents a particular marking of the T-BPN. Furthermore, each macro-state integrates a set of timed micro-states where each one corresponds to a particular clock value associated to each transition enabled by the marking.

A symbolic method can be used to relieve the above constraints, by taking the advantage of Binary Decision Diagrams (BDDs) capacity to represent a large set of encoded data with small data structure. The BDD structure is initially used to generate Petri net state space by Pasteur et al. [10]. This work develops an algorithm for PN state space exploration by encoding each PN place using a binary variable. Taking the advantage of the BDD compact representation, the exploration of reachable states is done within hours [13]. With the same objective, [11] introduced more useful coding using the place invariants. However, the underlying logic is always based on Boolean variables. A Multi-Valued Decision Diagram (MDD) that is an extension of BDD, have been used by [9] to improve the running time of algorithm previously given to state-space exploration. However, the methods based on BDDs have been successfully applied to untimed Petri nets, but for the TPNs less progress has been made.

In this work, the key idea is to introduce firstly a symbolic approach to compute the markings, in discrete time, of Safe Time Petri Net (STPN) that is composed of Time Binary Petri Nets set. Next we introduce the temporal information to the BDD tools, and we develop an algorithm which allows us to model in each time slice the reachable markings in a small data structure with polynomial complexity space. The structure is so-called Time Reduced Ordered Binary Decision Diagrams (TROBDDs). The state

space is built by exploration of TROBDDs and its storage into stacks. The intended application concerns the Flight Rescheduling Problem (FRP) due to a reduction of available capacity in air traffic network. In air traffic management the time slice is 15 minutes and the planning horizon is one day for tactical decision. So, the time study is limited to 96 time slices, which makes practicable the use of a discrete-time model. Our point of view on the FRP is focused on the accessibility problem under capacity constraints floating in time. We model the itineraries set of rescheduled flights using the STPN. Therefore, a cost function is associated to TROBDDs model. Next one can develop an optimization algorithm based on optimization criterion in order to determine the optimal flight plan.

This paper is organized as follows; the section 2 describes the background of the time Petri nets tools and binary decision diagrams. The section 3 depicts the symbolic approach to compute the reachable makings. The Section 4 presents the modeling reachable makings through TROBDDs. The exploration of state space and the storage technic is presented in Section 5. In order to illustrate the application of our approach, the FRP is presented in Section 6. We report the experimental results and discussion in Section 7. Finally, the conclusion and the future work are presented in Section 8.

2. Backgrounds

2.1 Time Petri Net (TPN)

A Time Petri Net (TPN) [2] is a tuple $N = (P, T, Pre, Post, I_s)$, where P is a finite set of places; T is a finite set of transitions; $Pre: P \times T \rightarrow \mathbb{N}$ is the pre-incidence function that determines weighted arcs from places to transitions; $Post: P \times T \rightarrow \mathbb{N}$ is the post-incidence function that determines weighted arcs from transitions to places; $\mathbb{N}$ is the set of non-negative integers and $I_s: T \rightarrow \mathbb{N} \times \mathbb{N}$ is a function which affiliates to each transition a time interval. For $t_i \in T$, $I_s(t_i) = [a_i, b_i]$ such that $a_i, b_i \in \mathbb{N}$ and $a_i \leq b_i$; $0 \leq a_i < \infty$; $0 \leq b_i < \infty$.

A marked TPN is characterized by the couple (N, M_0), where N is a TPN and M_0 is the initial marking. The state of TPN is defined by a couple (M, I_T), where M is a marking and I_T is the vector which contains firing intervals of transitions.

A TPN is said bounded if the set of reachable marking from M_0 is a finite set. A TPN is said k-bounded if for any place in reachable marking from M_0 does not exceed k. A TPN is said *safe* if it is 1-bounded. In this paper, we focused on Safe TPN (STPN) which is constituted of a set of Time Binary Petri Nets (T-BPNs).

Definition 1(Time Binary Petri Net) **:** A Time Binary Petri Net (T-BPN) is a STPN where only one place among all places can be marked:

- $\forall\, p \in P, M(p) \in \{0,1\}$;
- $\forall\, p_i, p_j \in P$ if $M(p_i) = 1$ then $M(p_j) = 0$;
- $Pre: P \times T \rightarrow \{0,1\}$ and $Post: P \times T \rightarrow \{0,1\}$.

The following properties emanate from definition 1.

Property 1. Let P_i be the places set of T-BPN$_i$ (respectively P_j for T-BPN$_j$), then $P_i \cap P_j = \emptyset$.

Property 2. Let T_i be the transitions set of T-BPN$_i$ (respectively T_j for T-BPN$_j$), then $T_i \cap T_j = \emptyset$.

Property 3. $\forall t_{i,j} \in T_i$, Card $({}^\bullet t_{i,j}) \in \{0,1\}$ and Card $(t_{i,j}{}^\bullet) \in \{0,1\}$.

The properties 1 and 2 mean that each place or transition belongs to a single T-BPN. The property 3 ensures that each transition of a T-BPN has at most one input place and at most one output place. Indeed, the T-BPN is a *state graph* that includes one or many elementary paths.

Definition 2. We define some notations for the T-BPN as follows:

- If the T-BPNi is composed of a single elementary path, we note $p_{i,j}$ (respectively $t_{i,j}$) the place j (respectively the transition j) of T-BPNi;
- If the T-BPNi is composed at least of two elementary paths, we note by $p_{i,j}$ the common place between the various elementary paths and by $p_{i,j}^k$(respectively $t_{i,j}^k$) the places of path k (respectively the transitions of path k).

The connection transitions between the T-BPN (have several places of input and/or several places of output) are denoted by a single index.

Figure 1 shows a $STPN$, which comprised of four T-BPNs. Each T-BPN contains the following places: $P_1 = \{p_{1,1},\ p_{1,2}, p_{1,3}\}$,$P_2 = \{p_{2,1}, p_{2,2}\}$, $P_3 = \{p_{3,1}, p_{3,1}^1, p_{3,1}^2, p_{3,2}\}$ and $P_4 =$

$\{p_{4,1}\}$. Furthermore, the T-BPNs are related by the transitions t_1 and t_2.

In the rest of this paper, the methodology exhibited is convenient for any class of STPN, which comprises a set of T-BPNs.

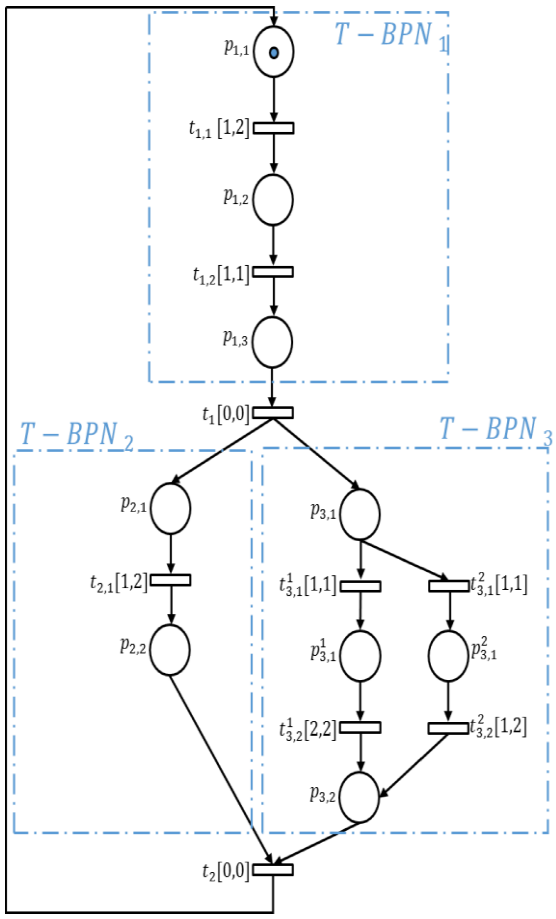


Figure 1. Example of STPN

2.2 Binary Decision Diagrams (BDDs)

Formally, a binary decision diagram is a directed acyclic graph with a set of vertices V, which has two types. Non-terminal vertices are indexed by a Boolean variable, x_i, denoted $index(v) = x_i$, $v \in V$. Each vertex x_i has two children: a left child and a right child, denoted respectively by $LeftChild(x_i)$, $RightChild(x_i) \in V$. Terminal vertices are indexed by the value 0 or 1 of the Boolean function. The $LeftChild(x_i)$ is bound to its parent by a dotted arrow where the value affected to x_i is 0, and the $RightChild(x_i)$ is bound to its parent by a continuous arrows where the value assigned to x_i is 1.

The representation technique of the Boolean functions using the BDD tool is based on the expansion of Shannon [3]. An *Ordered Binary Decision Diagram* (OBDD) is a BDD where the variables respect a topological order $x_1 < x_2 < \cdots < x_n$, whatever is the path of the root in the terminal vertices of BDD, [8]. A *Reduced Ordered Binary Decision Diagram* (ROBDD) is an OBDD where the size of the BDD is reduced by the application of reduction rules, such as the fusion of isomorphic sub-graphs [3, 14]. In the rest of the paper, when no confusion is possible, the size of the BDD corresponds to the number of non-terminal vertices.

3. Computing the STPN Markings using Symbolic Approach

In this section, we introduce a new symbolic approach to compute in discrete time the markings of STPN by a Boolean function.

Let M_P^τ be the power set of STPN places set at the time τ, with $Card(P) = n$. Let $\mathcal{P}(M_P^\tau)$ be the power set of M_P^τ. According to Boolean algebra of a power set [3], the system $(\mathcal{P}(M_P^\tau), \cup, \cap, \emptyset, M_P^\tau)$ is a Boolean algebra with $Card(\mathcal{P}(M_P^\tau)) = 2^{2^n}$, where $\emptyset$ the empty set and $\cup, \cap$ the union and the intersection laws.

Theorem of Stone **[12].** Any finite Boolean algebra is isomorphic to the Boolean algebra of subsets of some finite set.

Based on the theorem of Stone, the system $(\mathcal{P}(M_P^\tau), \cup, \cap, \emptyset, M_P^\tau)$ is isomorphic to the algebra of n-variables Boolean functions $(F^n(\mathcal{B}), +, \cdot, 0, 1)$, where $F^n(\mathcal{B})$ is the set of n-variables Boolean functions, $(+)$ and $(\cdot)$ the addition and the multiplication of n-variables Boolean functions. Also, $'0'$ is the function *zero* $(f_0(x_1, x_2 \ldots, x_n) = 0 \; \forall \, x_i \in \mathcal{B})$ and $'1'$ is the function *one* $(f_1(x_1, x_2 \ldots, x_n) = 1 \; \forall \, x_i \in \mathcal{B})$, (for more details see [4]).

Let $\psi^\tau : \mathcal{P}(M_P^\tau) \to \mathcal{B}$ be the *temporal symbolic* function, which allows to express at every time τ a marking in $\mathcal{P}(M_P^\tau)$ by a Boolean expression. We define the temporal symbolic function $\psi^\tau = \Upsilon^\tau \circ \mu^\tau$ by the composition of the *temporal ciphering function* $\mu^\tau : \mathcal{P}(M_P^\tau) \to \mathcal{B}^n$ and *temporal indicator* function $\Upsilon^\tau : \mathcal{B}^n \to \mathcal{B}$.

The temporal ciphering function consists in coding a marking m^τ by a summit $Z_{m^\tau} = (x_1, x_2, \ldots, x_n) \in \mathcal{B}^n$, with x_i the binary variable representing the place p_i. μ^τcan be represented as follows:

$$\mu^\tau : \begin{array}{l} \mathcal{P}(M_P^\tau) \longrightarrow \mathcal{B}^n \\ m^\tau \longmapsto Z_{m^\tau} = (x_1, x_2, \ldots, x_n) \end{array}$$

where:

$$\forall i \in 1,\dots n\ ;\ x_i = \begin{cases} 1 & \text{if } p_i \in m^\tau \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

The *temporal* indicator function Υ^τ takes as input Z_{m^τ} and returns a Boolean value as follows:

$$\Upsilon^\tau: \begin{matrix} \mathcal{B}^n \longrightarrow \mathcal{B} \\ Z_{m^\tau} \longmapsto \omega \end{matrix}$$

where:

$$\omega = \begin{cases} 1 & m^\tau \textit{ is a} \text{ reachable marking} \\ 0 & \text{otherwise.} \end{cases}$$

Let m_1^τ and m_2^τ be two markings of $\mathcal{P}(M_P^\tau)$ such as $\mu^\tau(m_1^\tau) = \mu^\tau(m_2^\tau)$. Since each place is coded by a binary variable, every marking is represented by a single argument. If $\mu^\tau(m_1^\tau) = \mu^\tau(m_2^\tau)$ then $m_1^\tau = m_2^\tau$, so μ^τ is injective. As a consequence ψ^τ is injective. On the other hand, the set of binary variables $\mathcal{B}$ contains the binary values $\{0,1\}$. At each time τ, there are: a non-reachable marking m_{na}^τ, such as $\Upsilon^\tau(m_{na}^\tau) = 0$, and a reachable marking m_a^τ, such as $\Upsilon^\tau(m_a^\tau) = 1$. So, ψ^τ is surjective. Consequently, the function ψ^τ is at the same time injective and surjective, so it is bijective.

The *temporal symbolic* function ψ^τ is a bijective function and thereby an isomorphism from $(\mathcal{P}(M_P^\tau),\cup,\cap,\emptyset,M_P^\tau)$ to $(F^n(\mathcal{B}),+,\cdot,0,1)$. Therefore, each marking in $\mathcal{P}(M_P^\tau)$ is represented by a Boolean function. Let m_1^τ, m_2^τ be two markings such as $\mu(m_1^\tau) = Z_{m_1^\tau}$ and $\mu(m_2^\tau) = Z_{m_2^\tau}$. The union (respectively the intersection) of $Z_{m_1^\tau}$ and $Z_{m_2^\tau}$ is represented by

$\Upsilon_{Z_{m_1^\tau} \cup Z_{m_2^\tau}} = \Upsilon_{Z_{m_1^\tau}} + \Upsilon_{Z_{m_2^\tau}}$(respectively $\Upsilon_{Z_{m_1^\tau} \cap Z_{m_2^\tau}} = \Upsilon_{Z_{m_1^\tau}} \cdot \Upsilon_{Z_{m_2^\tau}}$).

We consider the reachable markings of *STPN* in Figure1 at $\tau = 1$ corresponds to $m^1 = \{\{P_{1,1}\},\{P_{1,2}\}\}$. The image of m^1 by the ciphering function is represented by two arguments: (1,0,0,0,0,0,0,0,0,0) and $(0,1,0,0,0,0,0,0,0,0) \in \mathcal{B}^{10}$ which corresponds respectively to $\{p_{1,1}\}$ and $\{p_{1,2}\}$, with $Card(P) = 10$. The symbolic function of m^1 is $\psi^1 = x_{1,1} \cdot \bar{X}_{\{x_{1,1}\}} + x_{1,2} \cdot \bar{X}_{\{x_{1,2}\}}$. With $\bar{X}_{\{x_{i,j}\}}$ the Cartesian product of complement to all variables except $x_{i,j}$.

For example, $\bar{X}_{\{x_{1,1}\}} = \overline{x_{1,2}} \cdot \overline{x_{1,3}} \cdot \overline{x_{2,1}} \cdot \overline{x_{2,2}} \cdot \overline{x_{3,1}} \cdot \overline{x_{3,1}^1} \cdot \overline{x_{3,1}^2} \cdot \overline{x_{3,2}} \cdot \overline{x_{4,1}}$.

4. The Modelling of the Reachable Markings

4.1 Explicit representation of STPN markings by OBDDs

This section presents a new modeling method of the reachable markings of STPN by OBDDs in discrete time. By using the ordinary Depth-First Search (DFS) of OBDD, the different paths are explored. According to the value of terminal vertex, one can classify the paths explored in two sets, as follows:

- The set of paths, which ends with a terminal vertex indexed by '1', model the reachable markings, denoted by Spt_1.
- The set of paths, which ends with a terminal vertex indexed by '0' corresponds to unreachable markings denoted by Spt_0.

A path pt_1 modelizes a reachable marking ($pt_1 \in Spt_1$) of *S*TPN composed of k T-BPNs, is described by the proposition 1.

Proposition 1. Each path pt_1 in Spt_1 includes at most k non-terminal vertices, which each one has a right child.

Proof. Let a *S*TPN be consisted of k T-BPN. Besides, a vertex corresponds to a marked place that has only a right child in a given path pt_1. As a T-BPN has at most one marked place, a $pt_1 \in Spt_1$ of a STPN contains at most k non-terminal vertices whose each one has a right child.

Figure 2 shows the OBDD presenting the function $\psi^1 = x_{1,1} \cdot \bar{X}_{\{x_{1,1}\}} + x_{1,2} \cdot \bar{X}_{\{x_{1,2}\}}$ which models the marking $m^1 = \{\{P_{1,1}\},\{P_{1,2}\}\}$. For a simplified representation, only the variables of T-BPN$_1$ are considered regardless the T-BPN$_2$ and T-BPN$_3$ are not marked in $\tau = 1$. The full representation of OBDD contains two paths of Spt_1. Each one models a reachable marking. If we take the example of the path pt_1 modeling the reachable marking $\{p_{1,1}\}^*$; it has one vertex indexed by $x_{1,1}$ which has a right child, and the other vertices have each one a left child. The topological order used of variables is: $x_{1,1} < x_{1,2} < x_{1,3}$.

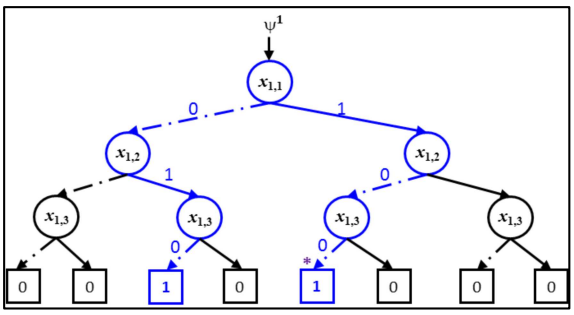


Figure 2. OBDD of the symbolic function ψ^1

4.2 TROBDDs construction

The full OBDD size is equal to $(2^n - 1)$ [6]. Therefore, the size of reserved memory space increases exponentially with the number of STPN places. Its complexity order is $O(2^n)$. Obviously, it is important to conceive an OBDD representation method with a reduced memory size.

In this paper, the behind construction of reduced OBDD at every time aims to represent only Spt_1. Therefore, one can only represent the variables storing the marked places.

Definition 3. The new structure proposed is so-called Time Reduced Ordered Binary Decision Diagrams ($TROBDD_s$) that model the reachable markings in discrete time. We denote by $TROBDD_\tau$ the modeling of reachable markings at the time τ.

The principle of a TROBDD$_\tau$ construction is illustrated by the algorithm in Figure 3; we begin by creating temporarily (can be deleted) the vertex. It is definitively created where does not annul the Boolean function and has a right child, i.e. For a current vertex, indexed by x_i, is definitively created only when $\psi^\tau(x_1 \dots x_i = 1)$ is not null.

The vertex x_i is definitively constructed and its children are temporarily constructed as shown in

$Construction_TROBDD_\tau(\psi^\tau(x_1 \dots x_n))$

Create temporarily the vertex x_1

For each variable x_i, $i = 1, \dots, n$ do

For each x_i temporarily Construction do

If $(\psi^\tau(x_1 \dots x_i = 1)! = 0)$

Create definitively the vertex x_i

Create temporarily $RightChild(x_i) = x_{i+1}$

If $(\psi^\tau(x_1 \dots x_i = 0)! = 0)$

Create temporarily $LeftChild(x_i) = x_{i+1}$

End If

Else If

Delete the vertex x_i

Create temporarily the vertex x_{i+1}

End If

End For

$i = i + 1$

End For

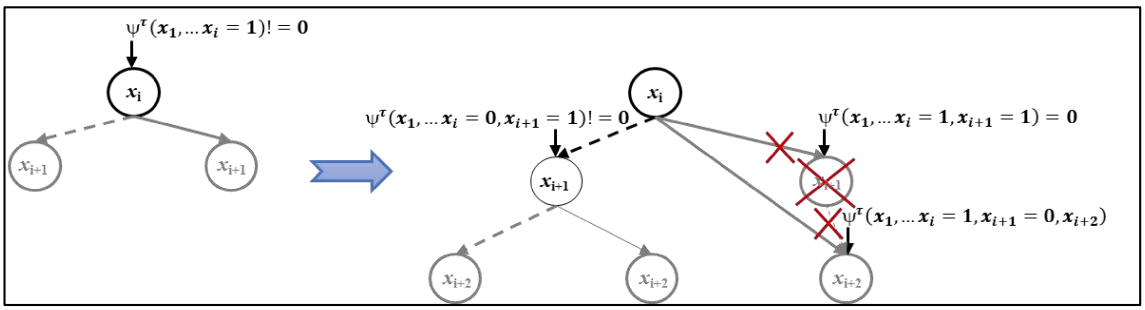
End.

Figure 3. Construction Algorithm of $TROBDD_\tau$

Figure 4.a. The second iteration consists in the calculation of the Boolean function where $x_{i+1} = 1$, and decides to delete or not each vertex x_{i+1}.

As shown in figure 4.b, one can eliminate the $RightChild(x_i) = x_{i+1}$ is done because the Boolean function is equal to zero. Then the $RightChild(x_i) = x_{i+2}$ is temporary constructed which corresponds to the left child of x_{i+1}. The $LeftChild(x_i) = x_{i+1}$ is definitively constructed where $\Psi^\tau(x_1, \dots, x_i = 0, x_{i+1} = 1)! = 0$. Then the children of x_{i+1} are temporarily created.

The application of reduction algorithm on the example of STPN (Figure 1) is illustrated in the



a) *b)*

Figure 4. Construction principle

following Figure 5. Every $TROBDD_\tau$ defines the reachable markings at τ. The depth first search in a given $TROBDD_\tau$ allows us to determine the different reachable markings.

***Proposition 2*:** The $TROBDD_\tau$ size resulting by the application of the algorithm1 has in the worst case a complexity order of $O(n)$; n is the number of STPN places.

Proof. Let us consider an STPN with n places. As in a given time τ, the TROBDD$_\tau$ models only the marked place. Then, in the worst case the TROBDD$_\tau$ size is equal to the number of STPN places.

5. TROBDDs Exploration

In this section, we discuss the states space generation by manipulating the $TROBDD_S$ structures, and the technique of storage.

5.1 The structure of vertex

The state space generation requires the respect of some constraints that must be recorded in each vertex of $TROBDD_S$; the temporal constraints to move from a vertex to another, and the precedence constraints that allows to define the next vertex. Therefore, the vertex in $TROBDD_S$ is declared as follows:

- ***index***: is the binary variable ($x_{i,j}$).
- $\boldsymbol{LRRT_{i,j}}$: is the Lower Residual Remaining Time ($LRRT$).
- $\boldsymbol{URRT_{i,j}}$: is the Upper Residual Remaining Time ($URRT$).
- ***Next*** : is the list of the next vertices of $x_{i,j}$.

For example, the vertex in $TROBDD_0$ (Figure 5) has the following information: $index(v) = x_{1,1}$, $LRRT_{1,1} = 1$, $URRT_{1,1} = 2$ and $Next = \{x_{1,2}\}$. The value of $LRRT$ and $URRT$ are updated at each incrementing of time .

5.2 The storage stacks and $TFGMEC$

We use two types of stacks to store the state space: The independents stacks ($Stack_z, z \in \mathbb{N}$) and a global stack $Stack_G$. Each independent stack stores in order the sates generated from the initial state to the final state. Each element of global stack points to an independent stack. Evidently, the state is determined by a depth-first search of $TROBDD_\tau$ during the run through $TROBDD_S$. A state at time τ is defined as follows: $E(\tau) = [x_{i_1,j_1}(\tau), \ldots, x_{i_k,j_m}(\tau)]$ where k is the number of T-BPN and m is the index of the marked place of T-BPNk.

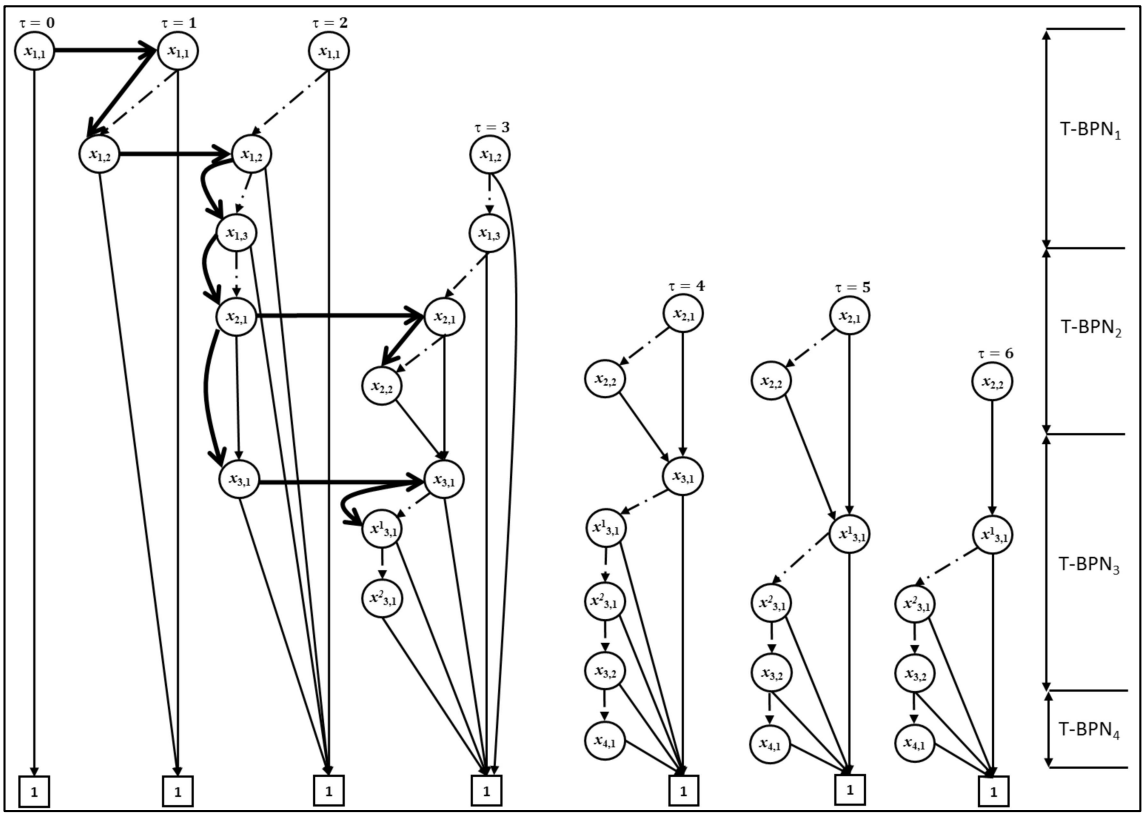


Figure 5. The $TROBDD_S$ of STPN in Figure 1

The modeling and analysis of real system should be taken into account the specification which float in time. For this specification types, one can introduce the Time Floating General Mutual Exclusion constraints $(TFGMEC)$ [1], which is a GMEC considered only in a time interval $[\tau_{min}, \tau_{max}]$. We introduce a new formulation of $TFGMEC$ suitable of the definition of state:

$$\vec{w}^T E_{[\tau_{min},\tau_{max}]} \leq \lambda \quad (3)$$

Where $\vec{w}$ is a binary vector and λ is an integer constant.

5.3 The generation and storage algorithm

The generation and storage algorithm consists of three steps, as is shown in Figure 6.

In the first step, from a current state $E(\tau)$, one can finds the set of indexes (X) which allows us to move to the next state. Then, either by the next $\left(x_{i,j}(\tau).Next\right)$ and / or by evolving the time $\left(x_{i,j}(\tau+1)\right)$, the set S of successors is calculated according to the value of the lower residual remaining time of each $x_{i,j}(\tau)$ in X.

The second step involves the determination of the successor's state E_s of $E(\tau)$. Once the set E_s is determined, an analysis is performed to eliminate the forbidden states that does not respect the $TFGMEC$. Therefore, the result is the set of legal states E_s which guarantees the desired behavior. The $(Card(E_s)-1)$ is the number of created stacks. The covered states are duplicated in current stack (step 3). Subsequently, the exploration of the global stack $(Stack_G)$ and the selection of the first independent stack are realized in step 4; note that the current state of first independent stack is not a final state.

As an example, we consider the $TROBDD_s$ in Figure 5. The DFS of $TROBDD_0$ give the initial state $[x_{1,1}(0)]$, that is stored in $Stack_1$, where its temporal constraints $LRRT_{1,1}=1$ and $URRT_{1,1}=2$. Obviously, the successor is $x_{1,1}(1)$ that leads a new state $[x_{1,1}(1)]$ generated from $TROBDD_1$ and stored in $Stack_1$ with an update of the temporal constraints that are $LRRT_{1,1}=0$ and $URRT_{1,1}=1$. By applying the step 2 of algorithm, the successor set is $S=\{x_{1,2}(1);\ x_{1,1}(2)\}$ that leads to two successors

Step 1: **Calculate the successors index:**

Let a current state $E(\tau)=[x_{i_1,j_1}(\tau),\ \dots, x_{i_n,j_m}(\tau)]$ stored in the $Stack_z$

1.1 Determine X $=\left\{x_{i,j}(\tau)\in E(\tau)|\ \delta=\min_{i,j}(LRTT_{i,j}), i=i_1\dots i_n, j=j_1\dots j_m\right\}$

$S=\emptyset$; $E_S=\emptyset$

1.2 For each $x_{i,j}(\tau)\in$ X

If $\delta=0$

If $URTT_{i,j}=0$ then $S=S\cup\{x_{i,j}(\tau).Next\}$

Else then $S=S\cup\{x_{i,j}(\tau).Next\}\cup\{x_{i,j}(\tau+1)\}$

Else $S=S\cup\{x_{i,j}(\tau+1)\}$

Step 2 : **Calculate the successor states of the trajectory and update the list of trajectories to be analyzed.**

2.1. Generate the set $E_s=\{E_1,E_2,\dots,E_{|S|}\}$ of successor states of the current state $E(\tau)$.

2.2. Eliminate the states that not respecting the $TFGMEC$ $\vec{w}^T E_{[\tau_{min},\tau_{max}]}\leq\lambda$

For each $E_r\in E_s, r=1,\dots,|s|$

If $\sum_{i,j}x_{i,j}(\tau)>\lambda$ then // $\tau\in[\tau_{min},\tau_{max}]$

$E_s=E_s\setminus E_r$,

Step 3: **Update the independents stacks**

If $E_s=\emptyset$ go to Step 4

Else

Update the $Stack_z$:

Choose $E_i\in E_s$

$Stack_z=Stack_z\cup\{E_i\}$

$E_s=E_s\setminus E_i$

While $E_s\neq\emptyset$

Create a new independent stack and duplicate the covered states in $Stack_z\setminus\{E_i\}$

Added the successor state $E_r\in E_s$.

Step 4 : **Determine the next independent stack to be analyzed.**

Explore the $Stack_G$ and select the first independent stack which the current state is not a final state.

If such an independent stack exists, go to Step 1.

Figure 6. Algorithm for generation and storage of state space

states. The state $[x_{1,2}(1)]$ is stored in $Stack_1$ and $[x_{1,1}(2)]$ is also stored in a new stack $Stack_2$. The predecessors states of $[x_{1,2}(1)]$ and $[x_{1,1}(2)]$ are the same as shown in Figure 7. Next, in the step 4 the $Stack_1$ is selected to carry the generating and the storing procedure. When we reach the state $[x_{2,1}(3), x_{3,1}(3)]$, in $Stack_1$ the successors of each one index are:

$S\left(x_{2,1}(3)\right) = \{x_{2,2}(3); x_{2,1}(4)\}$ and
$S\left(x_{3,1}(3)\right) = \{x^1_{3,1}(3); x^2_{3,1}(3)\}$.

The exploration step of $TROBDD_s$ generates the set of successors of $[x_{2,1}(3), x_{3,1}(3)]$:

$$E_s = \begin{Bmatrix} [x_{2,2}(3), x^1_{3,1}(3)]; & [x_{2,2}(3), x^2_{3,1}(3)]; \\ [x_{2,1}(4), x^1_{3,1}(4)]; & [x_{2,1}(4), x^2_{3,1}(4)] \end{Bmatrix},$$

Thus, one can apply the following $TFGMEC$:

$$\left(x_{2,1}(\tau) + x_{2,2}(\tau) + x^2_{3,1}(\tau)\right)_{[3,6]} \leq 1.$$

Therefore, the states $[x_{2,2}(3), x^2_{3,1}(3)]$ and $[x_{2,1}(4), x^2_{3,1}(4)]$ are forbidden from E_s and a new stack ($Stack_3$) is created.

The $TROBDD_s$ exploration stored in $stack_1$ is shown with bold arrows (i.e. see Figure 5).

6. Application to Air Traffic Flow Management Problem

The worldwide demand of air traffic is expanding that has continued to grow. However the capacity of Air Traffic Network (ATN) elements such as the airports, sectors and itineraries are not changed. Moreover, the ATN capacity is reduced for any adverse weather conditions and makes some scheduled flight plan unrealizable.

In this application, we focus on the flights before their take-off (uncommitted flight), where the scheduled flight plan is affected due to adverse weather condition and become unrealizable. Therefore, the Flight Rescheduling Problem (FRP) is involved in order to minimize the impact of unforeseen disruptions on the schedule planning. This is an accessibility control problem under safety constraints with floating temporal windows through time caused by adverse weather condition. In this context, we propose an optimization algorithm to calculate a new optimal flight plan by minimization the cost function.

6.1 Notations and modeling

6.1.1 Notations

The flights rescheduling problem includes a set of flights $F = \{1, 2 \ldots |F|\}$ which uses a set of ATN during a time window (a day for example). Each flight has a departure airport d_f, $f \in F$, of the set of airports $D = \{d_1, d_2 \ldots d_{|D|}\}$ and a landing airport a_f, $f \in F$, of the set of destination airports $A = \{a_1, a_2 \ldots a_{|A|}\}$. A set of air sectors which will be used by the flights is denoted by $S = \{s_1, s_2, \ldots s_{|S|}\}$. The itinerary for the flights that connects the departure to the arrival airport are numbered and defined by the set $I = \{1, 2 \ldots |I|\}$.

Definition 4: A Flight Itinerary Set (FIS_f) of flight $f \in F$ is the succession of ATN elements according to the flight time constraint of each one. We denoted it by $FIS_f = \{\Pi^i_f \mid f \in F, i \in I\}$, with $\Pi^i_f = d_{i,\mathrm{d}}(\theta_{\mathrm{d}}), s_{i,1}(\theta_{\mathrm{l}1}), \ldots \mathrm{s}_{i,\mathrm{k}'}(\theta_{\mathrm{lk}'}), \mathrm{a}_{i,\mathrm{a}}(0)$. For example, $s_{i,1}(\theta_{\mathrm{l}1})$ models the flight f following the itinerary i by overflying the sector s_1, and needs $\theta_{\mathrm{l}1}$ time unit. The available capacity of each ATN element $e_k \in D \cup S \cup A$ at the time interval $[\tau_1, \tau_2]$, is denoted by $Cap(e_k, [\tau_1, \tau_2])$, and expressed by a $TFGMEC$.

We admit that all the flights can have the similar flight time slice. We suppose that the flight duration for a given sector with an airport takes one time slice while it takes twice as long if the sector hasn't an airport. Note that a scheduled

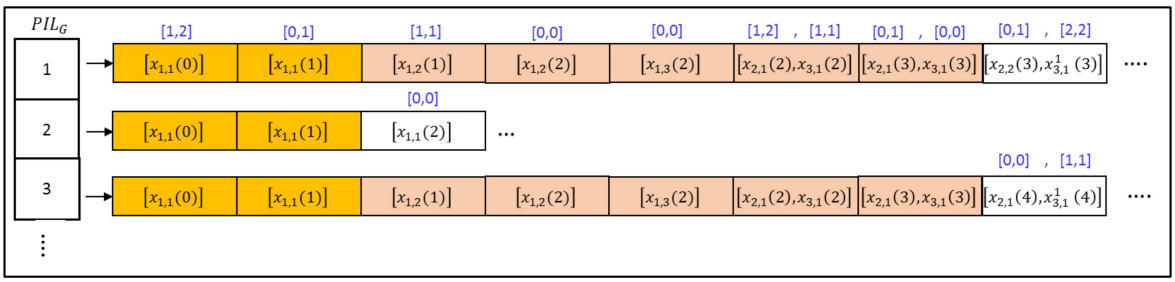


Figure 7. Example of storage in stacks

flight plan is considered as the shortest path connecting two airports.

The Figure 8 depicts an example of air traffic networks, which is composed of three departures airports (d_1, d_2 and d_2), an arrival airport a_1 and 19 sectors. The arcs connecting a pair of airports are the possible itineraries for a given flight, where their number is presented by a circle. We consider three flights, which each one departs from an airport, and have a common destination. For instance, the flight number $'1'$ from the airport d_1 can follow one of three itineraries of $FIS_1 = \{\Pi_1^1, \Pi_1^2, \Pi_1^3\}$. Moreover, the scheduled flight itineraries set is $\{\Pi_1^2, \Pi_2^5, \Pi_3^8\}$, where their itineraries are delineated by dotted lines.

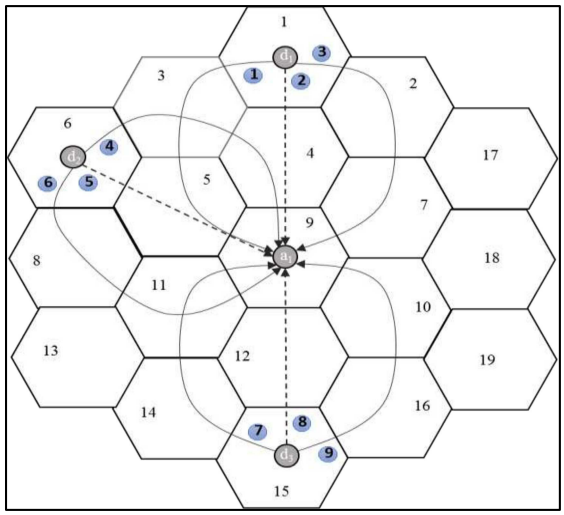


Figure 8. Example of air traffic network

6.1.2 Modeling a FIS by STPN

The key idea is to consider the air traffic system as a discrete events system for the flight rescheduling problem. We model all flights itinerary set; $FIS = \bigcup_f FIS_f$ by a STPN where each FIS_f is modeled by a T-BPN, noted T-BPN$_f$.

Relevant to the elements notation of a STPN compound of a T-BPN set (definition2), the places indicate the plane location in ATN, are denoted as follows:

- p_{f,d_j}: the departure airport d_j of flight f;
- p^i_{f,s_j}: the flight f crossing the sector s_j following the itinerary i;
- p_{f,a_j}: the arrival airport a_j of flight f.

The temporal constraint associated to the transitions indicates either the delay time tolerated by the flight in the departure airport $\left(t^{i}_{f,d_j}\right)$, or the time for a flight to overfly a sector $\left(t^{i}_{f,s_j}\right)$.

The Figure 9 illustrates the STPN modeling the $FIS_1 \cup FIS_2 \cup FIS_3$ of the air traffic example in the Figure 8. The T-BPNs model the FIS_1, FIS_2 and FIS_3. The maximal delay tolerated for the three flights before their cancellation is considered of 2 t.s (time slice). For example, for the flight 1, the necessary time following the itinerary Π_1^2 is 1 t.s to overfly the sector s_1, 2 t.s

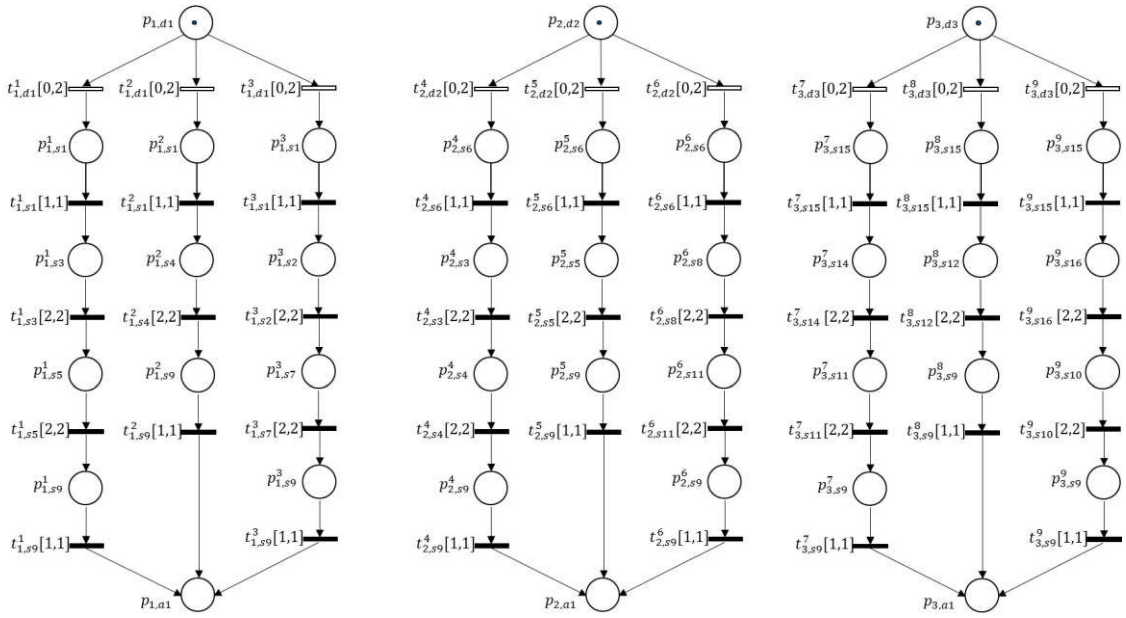


Figure 9. STPN modelling the *FIS*

to cross the sector s_4 and a time slice to overfly the sector s_9 in order to reach at the airport a_1.

Before the take-off of three flights an adverse weather condition affected the sectors s_4, s_5 and s_{12} in time interval [2,4] when the scheduled flights cross the sectors and reduce its capacity to zero: $Cap(s_4,[2,4]) = Cap(s_5,[2,4]) = Cap(s_{12},[2,4]) = 0$. Then, the rescheduling is unavoidable to decrease the perturbation impact on the scheduled flights. The rescheduled flight plan will respect the new constraints that is formulated by TFGMEC as follows:

$$\left(x^2_{1,s_4}(\tau) + x^4_{2,s_4}(\tau) + x^1_{1,s_5}(\tau) + x^5_{2,s_5}(\tau) + x^8_{3,s_{12}}(\tau)\right)_{[2,4]} \leq 0$$

6.2 TROBDD of the set of flight plans and optimal rescheduled flight plan

This section describes how the $TROBDD_S$ structure, the generation and storage approaches are defined in flight rescheduling problem.

The admissible markings in discrete time modeled by a $TROBDD_S$ structure allows us to record at every time the possible geographical locations of the rescheduled flights in a small data structure. The $TROBDD_\tau$ size is proportional to the number of possible locations of rescheduled flights at time τ (proposition 2).

The algorithm of generation and storage of state space takes into account a given $TFGMEC$ and allows to determine the set of rescheduled flights plans regarding the constraints capacity. Each independent stack memorizes a possible rescheduled flight plan. The rescheduled flight plan is a flight itinerary Π_f^i, with a fixed take-off time. To calculate its cost, we associate a cost structure to $TROBDD_S$ such as the penalty fee of ground delay and the cost of consumed kerosene (for more details see our previous work [7]). In the course of the generation and storage, the addition of a new state from E_s (step 3) is accompanied by the update of its economic performance by adding the cost associated with the recently integrated state. The cost associated to the final state is the total cost of the rescheduled flight plan.

To determine the optimal flight plan among all rescheduled flights plans in a minimal CPU time, we propose an optimization criterion which allows us to find the optimal plan without generating all solutions (see proposition 3).

Proposition 3. The economic analysis of air traffic show that the ground delay is negligible to airlines compared to the consumed kerosene cost during the flight. Indeed, for the reschedule flight plan it is better to delay the aircraft on the ground instead of changing its initial itinerary which is longer. Therefore, the generation and storage algorithm explores in the first time the solutions with the minimum flight distance, if the generation doesn't lead to a final state, then the second criterion that ground delay is adopted. Otherwise, we look for a flight plan solution with longer distance.

Remark. Based on the proposition 3, we note that it is not necessary to generate all the solutions. Once a solution is found, it is systematically the optimal.

7. Numerical Application and Discussion

In this section we expose the results of a set of computational experiments. We compare the efficiency of the $TROBDD_S$ structure to our previous TD-RG structure in [7]. Also, the performances calculation of the algorithm determines the optimal rescheduled flight plan.

For this purpose, we distribute equally each instance $|F|$ between the three departure airports presented in the Figure 8. For each case, we increase the flight number at every airport by 10 as indicated by the first column of Table 1. The maximal ground delay time is based on the profitability function [15]. The flight annulation takes 4 hours in the ground delay. Given that the time slice in air traffic is 15 minutes, the study time is equal to 22 time slices; where the tolerated ground delay is $\theta_d = 16$ time slices and the longest flight time takes 6 t.s.

Before the take-off of flights, an adverse weather condition, reduces the capacity of some sectors and makes the scheduled flight plan infeasible. As shown in column 2, the capacity constraints in the specific time interval are formulated by the $TFGMEC$. For example, the capacity of sector s_5 where the flights departures from d_1 $\left(f = 1 \ldots \frac{|F|}{3}\right)$ and d_2 $\left(f = \frac{|F|}{3} + 1 \ldots 2 \times \frac{|F|}{3}\right)$ is reduced to one-fifteenth of the scheduled flights through the time interval $[1, 16]$ that is formulated by the first $TFGMEC$.

Table 1. Computational results

$\|F\|$	TFGMEC	Structure size		FRP
		TD-RG nodes,[7]	TROBDDs size	CPU
15	$\sum_{f=1}^{\frac{\|F\|}{3}}\left(x_{f,s_5}^2(\tau)\right)_{[1,16]}+\sum_{f=\frac{\|F\|}{3}+1}^{2\times\frac{\|F\|}{3}}\left(x_{f,s_5}^4(\tau)\right)_{[1,16]}\leq\frac{\|F\|}{15}$	$1{,}81\ 10^2$	3555	0.36
45		$2{,}37\ 10^5$	10665	1.26
75		$1{,}41\ 10^8$	17775	2.46
90	$\sum_{f=1}^{\frac{\|F\|}{3}}\left(x_{f,s_4}^2(\tau)\right)_{[2,14]}+\sum_{f=\frac{\|F\|}{3}+1}^{2\times\frac{\|F\|}{3}}\left(x_{f,s_4}^4(\tau)\right)_{[2,14]}\leq\frac{\|F\|}{15}$	$3{,}02\ 10^{15}$	21330	2.62
105		$9{,}19\ 10^{21}$	24885	3.56
135	$\sum_{f=2\times\frac{\|F\|}{3}+1}^{\|F\|}\left(x_{f,s_{12}}^8(\tau)\right)_{[0,8]}\leq 0$	$7{,}87\ 10^{34}$	31995	5.05
165		$6{,}02\ 10^{68}$	39105	6.36
195	$\sum_{f=2\times\frac{\|F\|}{3}+1}^{\|F\|}\left(x_{f,s_{12}}^8(\tau)\right)_{[10,14]}\leq 0$	$4{,}09\ 10^{109}$	46215	7.39

The adverse weather condition also annul the capacity of the sector s_{12} in the interval of $[0,8]\cup[10,14]$, defined by the last two $TFGMEC$.

For each instance, the column 4 reports the $TROBDD_S$ size by the application of the construction algorithm presented in Figure 3. We can notice that the reduction rules allow to reduce considerably the $TROBDD_S$ size. More than 80 % of vertices are eliminated for all instances. The new $TROBDD_S$ structure leads to decreases drastically the size of memory space, compared to TD-RG structure that has previously been proposed [7], as shown in Table 1.

Besides, the $TROBDD_S$ modeled in a small data structure leads necessarily to an efficient manipulation to generate the optimal solution. The exploration of the optimal flight plan from the model $TROBDD_S$ is implemented in C++ under Windows 7 and the tests were realized on a PC with 2.1 GHz processor speed and 8Go of RAM. The last column of table I show the performance of the generation algorithm based on the optimization criterion to determine the optimal solution. For example, the optimization algorithm solves a flight rescheduling problem for 195 flights modeled by a $TROBDD_S$ with the size of 46215 vertices, with a time resolution less than 8 seconds.

8. Conclusion

In this paper a new approach so-called Time Reduced Ordered Binary Decision Diagrams ($TROBDDs$) is proposed which permit us to model in a small data the reachable markings in discrete time. It is founded on the decomposition of a Safe Time Petri Net (STPN) to a time binary Petri nets (T-BPN) set. Our new structure complexity order is polynomial compared to the initial size that is exponential. Indeed, a new technique is presented for the generation and storage of the state space of a STPN.

The application of our results has envisaged the Flight Rescheduling Problem (FRP). The set of rescheduled flights itineraries are modeled by a STPN compound of T-BPNs. Each one model the set of flight itineraries. Besides, a cost function is added to $TROBDD_S$, and an optimization criterion is adopted to generate the optimal flight plan. The computational results approve that our new approach is able to solve efficiently several instances of FRP. Also, one can deduce that a large set of reachable markings of STPN can be represented with a small data structure compared to our previous developed TD-RG structure.

Finally, to increase the applicability of the presented approach, our work prospects deals with the flight rescheduling problem for continues flights where the aircraft performs several flights on one day.

REFERENCES

1. ACHOUR, Z., N. REZG, **Time Floating General Mutual Exclusion Constraints**, Studies in Informatics and Control, vol. 16, no. 1, 2007, pp. 57-66.
2. BERTHOMIEU, B., F. VERNADAT, **State Class Constructions for Branching Analysis of Time Petri Nets,** Tools and Algorithms for the Construction and Analysis of Systems. Springer Berlin Heidelberg, 2003. pp. 442-457.
3. BRYANT, R.E, **Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams**, ACM Computing Surveys, vol. 24, no. 2, 1992, pp. 293-318.

4. BROWN, F. M, **Boolean Reasoning**, Dordrecht: Kluwer Academic, 1990.

5. CHATAIN, T., C. JARD, **Complete Finite Prefixes of Symbolic Unfoldings of Safe Time Petri Nets**, In Petri Nets and Other Models of Concurrency-ICATPN Springer Berlin Heidelberg, 2006, pp. 125-145.

6. FUJITA, M., Y. MATSUNAGA., T. KAKUDA, **On Variable Ordering of Binary Decision Diagrams for the Application of Multi-level Logic Synthesis**, in: Proceedings of the Conference on European Design Automation, 1991, pp. 50-54.

7. KAMMOUN, M. A., N. REZG, Z. ACHOUR, **New Approach for Air Traffic Management based on Control Theory**, International Journal of Production Research (IJPR), vol. 52, no.6, 2014, pp.1711-1727.

8. LIAW, H. T., C. S. LIN, **On the OBDD-Representation of General Boolean Functions**, IEEE Transactions on Computers, vol. 41(6) , 1992, pp. 661-664.

9. MINER, A. S., G. CIARDO, **Efficient Reachability Set Generation and Storage using Decision Diagrams**, Application and Theory of Petri Nets Springer Berlin Heidelberg, 1999, pp. 6-25.

10. PASTOR, E., O. ROIG., J. CORTADELLA., R. M. BADIA, **Petri Net Analysis using Boolean Manipulation**, Application and Theory of Petri Nets, Springer Berlin Heidelberg, 1994, pp. 416-435.

11. PASTOR, E., J. CORTADELLA, **Efficient Encoding Schemes for Symbolic Analysis of Petri Nets**, Proceedings of the Conference on Design, Automation and Test in Europe. IEEE Computer Society, 1998. pp. 790-795.

12. PASTOR, E., J. CORTADELLA., O. ROIG, **Symbolic Analysis of Bounded Petri Nets**, IEEE Transactions on Computers, vol. 50, no. 5, 2001, pp. 432-448.

13. ROIG, O., J. CORTADELLA., E. PASTOR, **Verification of Asynchronous Circuits by BDD-based Model Checking of Petri Nets**, In Application and Theory of Petri Nets, 1995, pp. 374-391.

14. RUDELL, R, **Dynamic Variable Ordering for Ordered Binary Decision Diagrams**, In Proceedings of IEEE/ACM International Conference on Computer-aided Design, 1993, pp. 42-47.

15. TOKTAS,B, **Addressing Capacity Uncertainty in Resource-Constrained Assignment Problems**, PhD Thesis, University of Washington, 2003.