UNIVERSITY OF MACEDONIA
POSTGRADUATE PROGRAMME
DEPARTMENT OF APPLIED INFORMATICS

MODELING – SIMULATION OF GREEN SUPPLY CHAINS IN AUTOMOTIVE INDUSTRIES

Thesis

by

© Apostolos Saroglou

Thessaloniki, December 2022

MODELING – SIMULATION OF GREEN SUPPLY CHAINS
IN AUTOMOTIVE INDUSTRIES

Apostolos Saroglou

B.Sc. in Mathematics, Auth, 2021

A thesis submitted of the requirements for the degree of Master of Science in Applied Informatics

Supervisor
Souravlas Stavros

Approved by the three-member committee on 01/03/2023

fullname 1	fullname 2	fullname 3
Souravlas Stavros	Sifaleras Angelos	Vergidis Konstantinos
.....	.....	.....

Apostolos Saroglou

.....

Abstract

The main goal of this thesis is to improve a proposed model (that was designed using conventional Petri Nets) using Colored Petri Nets. The proposed model simulates the implementation of green supply chains in a hypothetical automotive industry that manufactures four different types of automobiles. Some introductory concepts - keywords (such as what is a green supply chain) are introduced at the outset. Then, relevant research on the modeling and design of green supply chain is presented. In the main part (chapter 3), the colored Petri nets are analyzed, such as, for example, what are their main structural elements and some of them are listed and explained. The reasons for the proposed model's improvement are stated in paragraph 3.2. Section 3.3 describes in detail the new improved model. Finally, in heads 4 and 5, an attempt is made to put the new improved model into action. More specifically, the model is implemented in Python (some structural elements of the code are presented), and the program that is created then runs a random number of simulations. The output data is used to generate charts and perform statistical analyses.

Keywords: Green Supply Chains, Automotive Industry, Petri Nets, Colored Petri Nets, Colored Petri Net Optimized Model, Python

Acknowledgments

I am overwhelmed with humility and gratitude to all those who have assisted me in putting these ideas, which are far above the level of simplicity, into something concrete.

I'd like to express my heartfelt gratitude to my teacher, Souravlas Stavros, who also assisted me in conducting extensive research and learning about many new things. I am extremely grateful to him.

Any attempt, at any level, cannot be completed satisfactorily without the support and guidance of my parents and friends.

For this reason, I would like to thank my parents and my friends who helped me mentally to face the difficulties that i was experiencing during my studies.

Table of Contents

Abstract..... 4
Acknowledgments..... 4

1. Introduction..... 8
1.1. Petri Nets..... 8
1.2. Colored Petri Nets..... 9
1.3. Green Supply Chains..... 11
1.4. Green Supply Chain Management..... 13
1.5. Problem Outline..... 14

2. Related Work..... 22

3. Methodology..... 23
3.1. Expressions..... 23
3.1.1. Colored Tokens..... 23
3.1.2. Multifunctional Transitions..... 24
3.1.3. Guards..... 25
3.1.4. Places..... 26
3.2. Disadvantages of the Proposed Model - Advantages of the CPN Optimized Model..... 27
3.3. The CPN Optimized Model..... 28
3.3.1. Overview..... 28
3.3.2. Initial Markings..... 38
3.3.3. Reachability Graph – Deadlocks..... 41
3.3.4. Execution Example..... 43

4. Programming of the CPN Optimized Model in Python..... 46

5. Simulation Results..... 51

Conclusion - Future Work..... 57

Bibliography..... 57

List of Figures

Figure 1: A simple - Petri - Net – Model, Source: Adapted from [3, Fig. 1] 8
Figure 2. The regular Petri net model of three shared memory system, Source: Adapted from [1, Fig. 2.1] 10
Figure 3. The colored Petri net model of three shared memory system, Source: Adapted from [1, Fig. 2.2] 11
Figure 4. Advantages of a Green Supply Chain, Source: Adapted from [6, Fig. 14.5] 12
Figure 5. Green Supply Chain Management Practices, Source: Adapted from [8, Fig. 1] 14
Figure 6. Green Supply Chain Model for Automotive Industry, Source: Adapted from [9, Fig. 3] 19
Figure 7. The processing of raw material a, Source: Adapted from [9, Fig. 5] 21
Figure 8. A simple CPN Model 23
Figure 9. Two initialized discrete black tokens in place p_1 24
Figure 10. CPN Model status after t_1 fires 25
Figure 11. Addition of guards in the CPN Model example 26
Figure 12. The CPN Optimized Model 29
Figure 13. Initial Markings of the CPN Optimized Model 40
Figure 14. Reachability-Graph of the CPN Optimized Model 42
Figure 15. Execution Example - Step 2 44
Figure 16. Execution Example - Step 1 44
Figure 17. Execution Example - Step 3 45
Figure 18. Execution Example - Step 4 45
Figure 19. Class Material 46
Figure 20. Class Manufacturer 47
Figure 21. Class Automobile 47
Figure 22. Places - Lists 48
Figure 23. Function t1_t2_t3_t4() 49
Figure 24. Colored Petri Net Coding 50

List of Tables

Table 1. Differences between Traditional and Green Supply Chains, Source: Adapted from [7, p. 21] 13
Table 2. Places and Meanings of Figure 6, Source: Adapted from [9, Tbl. 2] 15
Table 3. Transition and Meaning of Figure 6, Source: Adapted from [9, Tbl. 3] .. 16
Table 4. Example of PEPA Models, Source: Adapted from [9, Tbl. 5] 20

Glossary

CPN : Colored Petri Net(s)
GSC : Green Supply Chain(s)
GSCM : Green Supply Chain Management
PEPA : Performance Evaluation Process Algebra

1. Introduction

1.1 Petri Nets

Petri Nets were invented by C.A Petri during the elaboration of his thesis in 1939. They are a powerful graphical formalism for the modelling and analysis of systems, especially for systems which exhibit synchronization and concurrency [1].
Petri Nets are mainly based on modelling systems and not on simulating them. However, Petri Nets can be simulated. Some fields of application include:

a) Production Systems (e.g., supply chains)
b) Communication Systems (e.g., communication protocols between networks)

To begin with, there are many different variations and applications of Petri Nets since they are used extensively worldwide. Although, in this thesis we will deal with Colored Petri Nets which will be defined and described later in this Section. Informally, in a Petri Net graph, we can detect three main types of shapes. The circles, which represent the places of the system, the lines which represent the transitions of the system and the directed input and output arcs of the transitions. To put it better, we will highlight the formal definition of the basic version of Petri Nets. A Petri Net Structure, we call it *C*, is an arranged set of four entities. Specifically[2, pp. 133–146], $C = (P,T,I,O)$ where:
$P = \{p_1,p_2,\ldots,p_n\}$ is a finite set of places or nodes, $n \geq 0$
$T = \{t_1,t_2,\ldots,t_m\}$ is a finite set of transitions, $m \geq 0$
(The sets P,T are disjoint)
$I : T \rightarrow R^{\infty}$ is an input function, which maps a transition to a bag of places
$O : T \rightarrow R^{\infty}$ is an output function, which maps a transition to a bag of places
(A bag is a generalization of sets that allows multiple occurrences of the same place – element)
For example, consider the Petri net graph below:

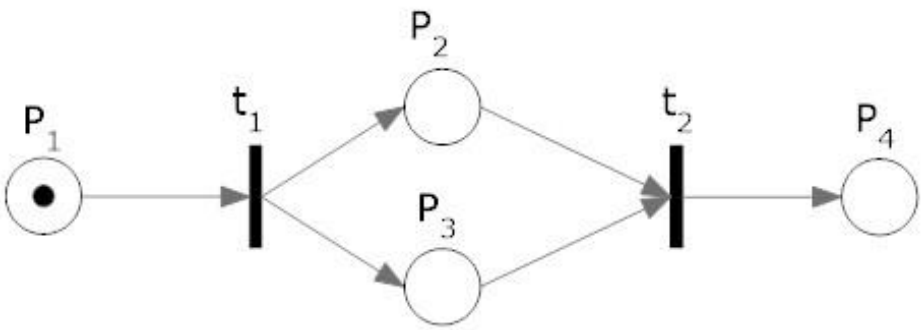


Figure 1: A simple - Petri - Net – Model
Source: Adapted from [3, Fig. 1]

By the definition of the Petri net structure, we have the following sets:

$C = \{P,T,I,O\}$
$P = \{p_1, p_2, p_3, p_4\}$
$T = \{t_1, t_2\}$
$I(t_1) = \{p_1\}$
$O(t_1) = \{p_2, p_3\}$
$I(t_2) = \{p_2, p_3\}$
$O(t_2) = \{p_4\}$

Furthermore, the previous definition suggests that a Petri net structure is static, but this is not true. A Petri net structure is dynamically modified by tokens. The tokens are "marks" and they are placed in the nodes of the network. When we initialize a Petri net model by placing tokens in nodes so that the system can start and firing transitions, we call that "initial token markings". The markings change each time depending on which transitions are activated. Activated is that transition for which each of its input places has at least as many tokens as the edges that are directed from the position to the transition. When a transition can be fired, tokens are removed from the input places and are transferred to the output places. The transfer of tokens from input places to output places is not a one-to-one process: For example, the number of tokens that a transition needs to fire is not necessarily the same with the number of output tokens after the transition fires. A PN model requires at least one active transition at a time, otherwise deadlock occurs.
However, if the designer of the model does not provide for the correct functionality of the model's execution, there is a chance of encountering malfunctions such as deadlocks and crashes in the system. A Petri net model is deadlocked when there are no transitions enabled.

Respectively, it is a common collide that there are transition crashes in a Petri Net model. Two transitions are in crash when they are both activated and their input sets have at least one common position, which has fewer tokens than required to fire both transitions. For this reason, it is very important to thoroughly check the execution and functionality of the model. It should be noted that tools such as reachability graphs help so that someone can understand if there are deadlocks in their system.

1.2 Colored Petri Nets

As it was mentioned in Section 1.1, Colored Petri Nets will be used as the main tool in this thesis. Colored Petri Nets are a special extension of the conventional Petri Net models. More specifically, the difference lies in the fact that the tokens have attributes (called colors) and these attributes can represent a variety of information related to tokens. For example, a token can contain information such as weight thus representing the entity – human being in a system. However, it can also have any characteristics such as (weight, height, skin color, age) thus describing the entity as detailed as possible. Colored Petri Nets are commonly used to simplify the complexity of a conventional Petri Net model. For instance, if a system models a process that is repeated because of the different entities participating in the system, it is possible to

simplify the system by using the process once and putting colors on the tokens. *Sadegh Ekrami in his thesis "Bindings in Colored Petri Nets" (2014)* gives a good example of transforming a conventional Petri Net model into Colored Petri Net model.
Particularly, he describes the example as it is shown in brackets [1, p. 11] and Figures 2,3 below:

> "*In Figure 2 the three subnets representing the processors in Figure 3 are "folded" into one subnet, while the processors are represented by three colored tokens which are quite independent from each other with the exception of accessing memory. If more than one processor accesses memory at the same time, the original priorities are taken into account*"

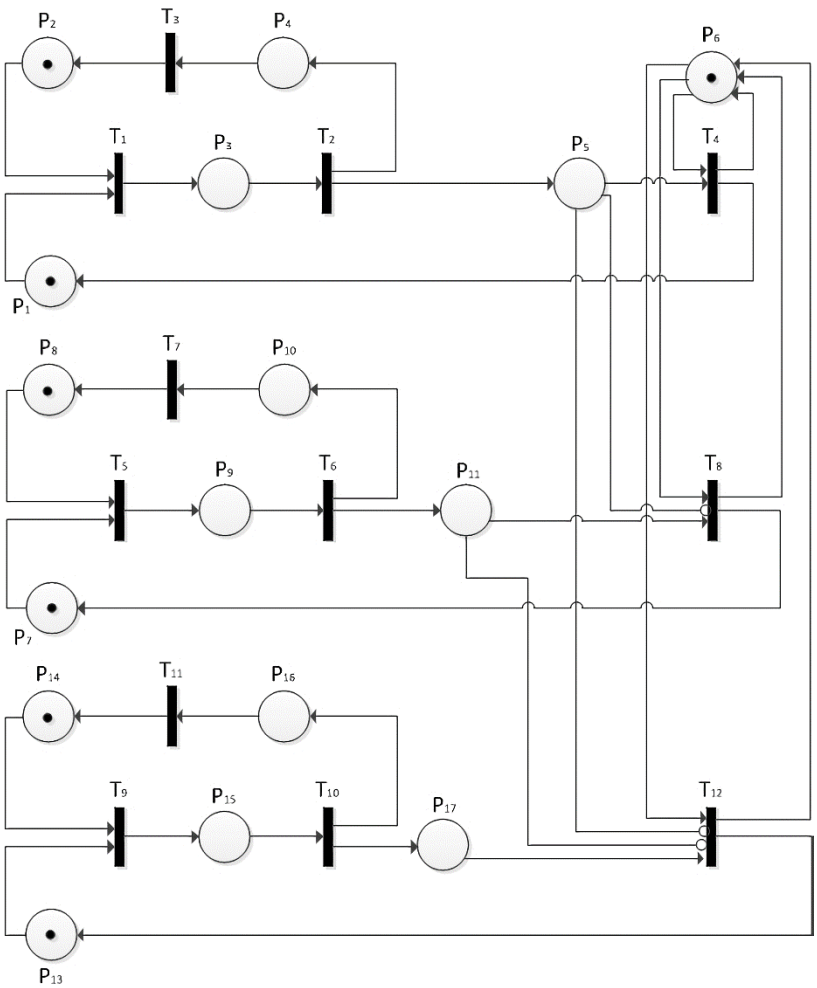


Figure 2. The regular Petri net model of three shared memory system
Source: Adapted from [1, Fig. 2.1]

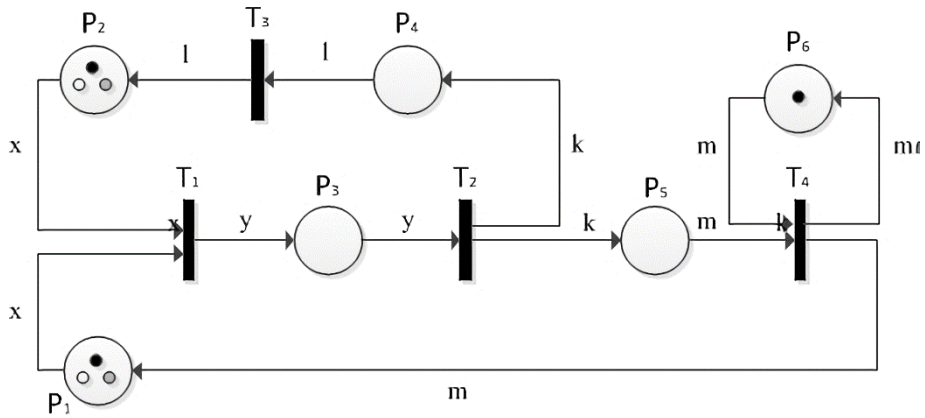


Figure 3. The colored Petri net model of three shared memory system
Source: Adapted from [1, Fig. 2.2]

Formally speaking, a Colored Petri Net satisfies the original definition of Petri Net Structure plus the functions – operations that are listed below[1, p. 13], [4]:

- **Σ** is a set of colors. This set contains all possible colors, operations and functions used within the Colored Petri Net (e.g., Σ = {*red, green, blue*})

- ***C*** is a color function. It maps places in *P* into colors in Σ.
  For example:

$$\forall\, c\ \in C: m(p) = \begin{cases} 1,\ if\ c = red; \\ 2,\ if\ c = green; \\ 3, if\ c = blue; \\ 0, otherwise \end{cases}$$

  If the place p has two red tokens and blue token then the marking can be written as: 1red + 1red + 3blue

- ***G*** is a guard function. It maps each transition $t \in T$ to a guard expression g. The output of the guard expression evaluates if the colored tokens satisfy the guard expression. It returns True if is satisfied and False if it is not satisfied. If the return is True then the transition can be fired and the tokens that satisfied the expression are removed. If the return is False, the transition cannot be fired

1.3 Green Supply Chains

An important factor and basis for this thesis is the idea of "*green supply chains*", as will be mentioned in the outline of the problem in Section 1.5. Below we will give the definition of green supply chains:

"GSC is a value chain that integrates environmental activities into supply chain practices like reducing pollution and wastes, reducing consumption of resources and energy used, and finally extend the life of products through recycling or remanufacturing" [5, p. 443]

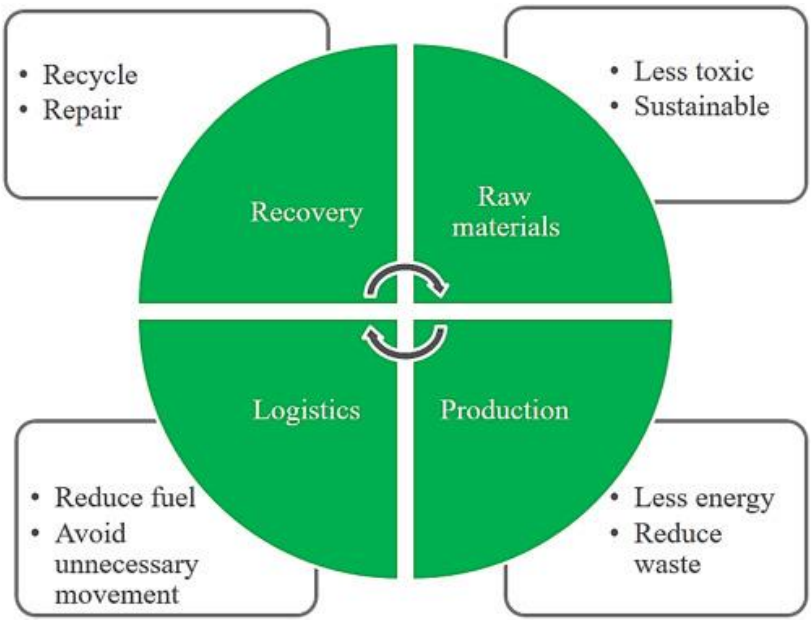


Figure 4. Advantages of a Green Supply Chain
Source: Adapted from [6, Fig. 14.5]

Green supply chains are a hot topic that concerns not only global industries and economies but much more the sustainability of our planet. Thus, it is imperative nowadays for companies to adopt green supply chain practices because they offer greater efficiency and ecology. However, factors such as lack of knowledge and techniques, government initiatives and high costs hinder the implementation and adoption of green supply chains as a mainstream practice in industries. The main difference between a green supply chain and a traditional one is that a green supply chain system can be visualized as a closed loop including reversed processes (reversed logistics) like recycling, reuse. Instead, a traditional supply chain is one-way production line. It is called one-way because it has a start meaning that raw materials are converted to products and an end meaning that the consumer throws the product in waste. The Table 1 below also shows some other important differences between green supply chains and traditional ones.

Characteristics	**Green Supply Chain**	**Traditional Supply Chain**

Aims and Value	Economical & environmental	Economical
The ideal environment	Environmental impact is low	Environmental impact is high
Suppliers standard Choice	Price and relationships on long-term	Price and relationships on short-term
Cost & price	High costs and High prices	High costs and Low prices
Speed and Flexibility	Low	High

Table 1. Differences between Traditional and Green Supply Chains
Source: Adapted from [7, p. 21]

Implementing a green supply chain, it has many pros like:

- Effective management of suppliers
- Using modern techniques of communication among chain colleagues
- Transparency of the supply chain
- Large investments and risks are shared among partners in the chain
- Increased quality of control in production
- Increased sales and revenue
- Beneficial uses for waste.

1.4 Green Supply Chain Management

Green Supply Chain Management is based on both environmental management and supply chain management. GSCM adopts practices that follow the traditional way of a supply chain in a more environmental-friendly way. There are plenty green supply chain practices. The main practices are among others:

- Green Design
- Green Manufacturing
- Green purchasing / consumption
- Products' end of life management (e.g., Recycle Centers)
- Green logistics (e.g., distribution, warehousing, transportation using biofuels)
- Green Marketing

The Figure below visualizes the green supply chain management and the life cycle of products:

Figure 5. Green Supply Chain Management Practices
Source: Adapted from [8, Fig. 1]

1.5 Problem Outline

This thesis focuses on the importance of green supply chains by optimizing and simulating a proposed green supply chain model with Colored Petri Nets. More specifically, the proposed green supply chain is part of the research work "Hierarchical Structure of a green supply chain" which was carried out in *(2021)* by *Jie Ding , Xiao Chen, Hui Sun , Wei Yan , Huixing Fang* [9]. The research is based on the fact that a supply chain consists of many sub-supply chains that interact with each other. Thus, the researchers conclude that there is a hierarchical structure in supply chains. Practically, they develop a green supply chain model upon an automotive industry. Because of the fact that the model consists of four concurrent and depended supply sub-chains for the production of four automobiles (type 1,2,3,4) , they apply compositional modeling using algorithms such as PEPA models to extract the sequence of events that occurs for each automobile of a sub-chain. To understand it better, let's take a better look at the proposed model shown in Figure 6. Table 2 below shows the places of the proposed model and their meanings. Furthermore, Table 3 highlights the transitions and their meanings.

Table 2. Places and Meanings of Figure 6
Source: Adapted from [9, Tbl. 2]

PLACE	MEANING
p_1	Raw material a
p_2	Raw material b
p_3	Provincial supplier 1
p_4	Provincial supplier 2
p_5	Municipal supplier 1
p_6	Municipal supplier 2
p_7	Raw material d
p_8	Raw material c
p_9	County supplier 1
p_{10}	County supplier 2
p_{11}	County supplier 3
p_{12}	Township supplier 1
p_{13}	Township supplier 2
p_{14}	Township supplier 3
p_{15}	Township supplier 4
p_{16}	Manufacturer 1
p_{17}	Manufacturer 2
p_{18}	Manufacturer 3
p_{19}	Manufacturer 4
p_{20}	Finished automobile 1
p_{21}	Finished automobile 2
p_{22}	Finished automobile 3
p_{23}	Finished automobile 4
p_{24}	Distributor 1
p_{25}	Distributor 2
p_{26}	Distributor 3
p_{27}	Distributor 4
p_{28}	Retailor 1
p_{29}	Retailor 2
p_{30}	Retailor 3
p_{31}	Retailor 4
p_{32}	Customer 1
p_{33}	Customer 2
p_{34}	Customer 3
p_{35}	Customer 4
p_{36}	Recycling center 1

p_{37}	Recycling center 2
p_{38}	Recycling center 3
p_{39}	Recycling center 4

Table 3. Transition and Meaning of Figure 6
Source: Adapted from [9, Tbl. 3]

TRANSITION	**MEANING**
t_1	Raw materials a and b are supplied by provincial suppliers 1 and 2
t_2	Raw materials are supplied from provincial supplier 1 to municipal supplier 1
t_3	Raw materials are supplied from provincial supplier 1 to county supplier 1
t_4	Raw materials are supplied from provincial supplier 2 to municipal supplier 2
t_5	Raw materials are supplied from municipal supplier 1 to county supplier 1
t_6	Raw material d and raw materials from municipal supplier 2 are supplied by county supplier 2 and 3
t_7	Raw material c and raw materials from county supplier 1 are supplied by township suppliers 1 and 2
t_8	Raw materials are supplied from county supplier 2 to township supplier 3
t_9	Manufacturer 3 receives raw materials from county supplier 2

t_{10}	Raw materials are supplied from county supplier 3 to township supplier 4
t_{11}	Manufacturer 1 receives raw materials from township supplier 1 Township supplier 2
t_{12}	Manufacturer 2 receives raw materials from township supplier 2
t_{13}	Manufacturer 3 receives raw materials from township supplier 3
t_{14}	Manufacturer 4 receives raw materials from township supplier 4
t_{15}	Manufacturer 4 produces auto-motive product 1
t_{16}	Manufacturer 2 produces auto-motive product 3
t_{17}	Manufacturer 3 produces auto-motive product 2
t_{18}	Manufacturer 1 produces auto-motive product 4
t_{19}	Auto-motive product 1 is sent to distributor 1
t_{20}	Auto-motive products 2 and 3 are send to distributors 2 and 3
t_{21}	Auto-motive product 4 is sent to distributor 4
t_{22}	Distributor 1 assigns auto-motive product to retailer 2
t_{23}	Distributor 2 assigns auto-motive product to retailer 1
t_{24}	Distributor 3 assigns auto-motive product to retailer 4

t_{25}	Distributor 4 assigns auto-motive product to retailer 3
t_{26}	Retailer 1 sells auto-motive product to customer 1
t_{27}	Retailers 2 and 3 sells auto-motive product to customers 2 and 3
t_{28}	Retailor 4 sells auto-motive product to customer 4
t_{29}	Customer 1 returns auto-motive product to recycling center 1
t_{30}	Customer 2 returns auto-motive product to recycling center 2
t_{31}	Customer 3 returns auto-motive product to recycling center 3 Customer 1
t_{32}	Customer 4 returns auto-motive product to recycling center 4 Customer 3
t_{33}	Recycling centers 1 and 2 sort, recycle and reuse
t_{34}	Recycling centers 3 and 4 sort, recycle and reuse

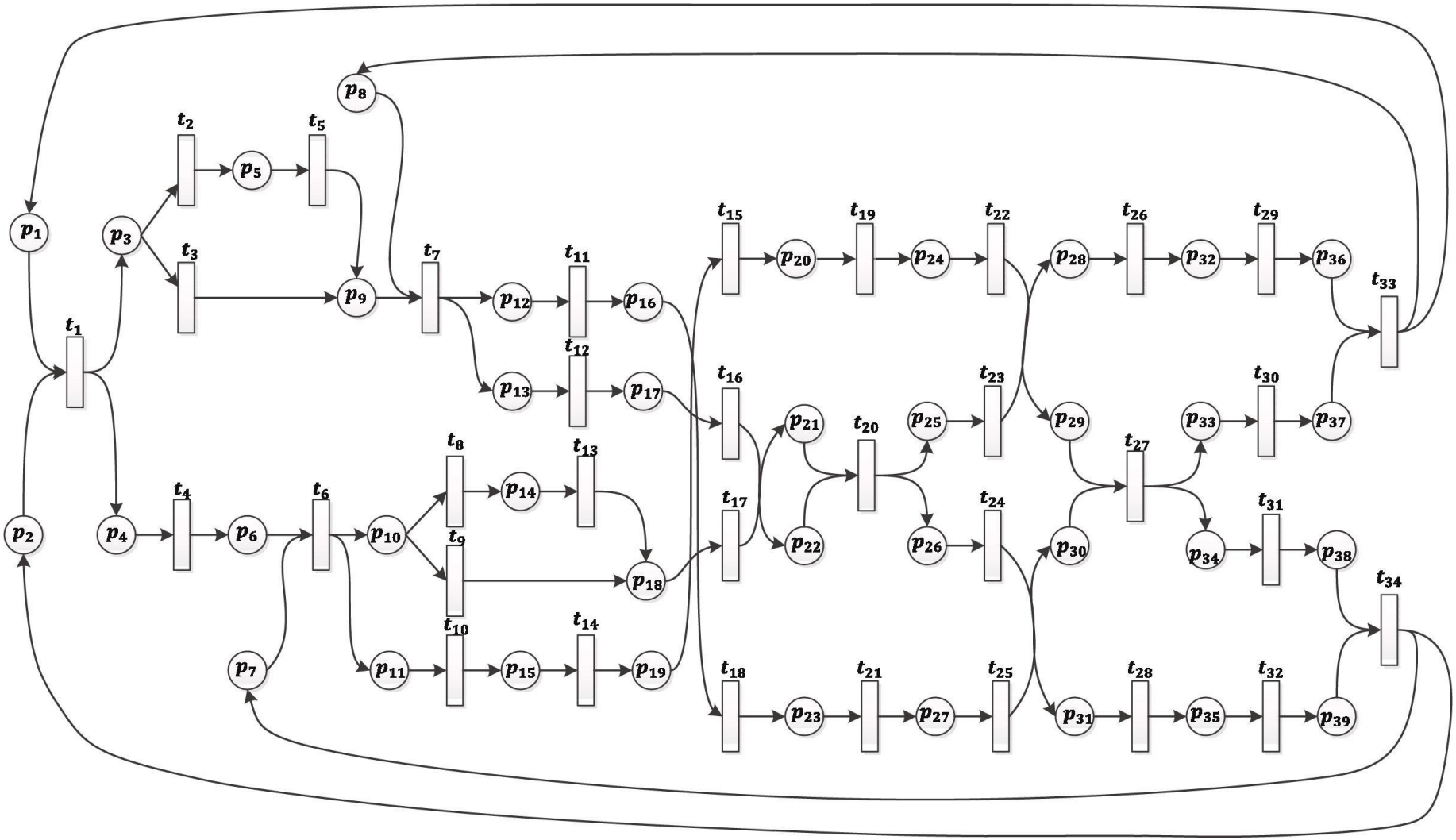


Figure 6. Green Supply Chain Model for Automotive Industry
Source: Adapted from [9, Fig. 3]

The researchers used the proposed Petri net model and mathematical algorithms to convert it into PEPA models. PEPA models will not concern us in this thesis , but briefly every PEPA Model corresponds to the different combination of processes of each raw material in the green supply chain system. Table 4 shows two PEPA Models. After the PEPA Models are extracted, we can draw conclusions for every raw material. Lastly, Figure 7 illustrates visually the traces of raw material a in PEPA Model 1.

Table 4. Example of PEPA Models
Source: Adapted from [9, Tbl. 5]

PEPA	**raw material a**	**raw material b**	**raw material c**	**raw material d**
Model 1	{p1, p3, p5, p9, p12, p16, p23, p27, p30, p33, p37}	{p2, p4, p6, p10, p14, p18, p21, p26, p31, p35, p39}	{p8, p13, p17, p22,p25,p28,p32, p36}	{p7, p11, p15, p19, p20, p24, p29, p34, p38}
Model 2	{p1, p3, p5, p9, p12, p16, p23, p27, p30, p33, p37}	{p2, p4, p6, p11, p15, p19, p20, p24, p29, p34, p38}	{p8, p13, p17, p22,p25,p28,p32, p36}	{p7, p10, p14, p18, p21, p26, p31, p35, p39}

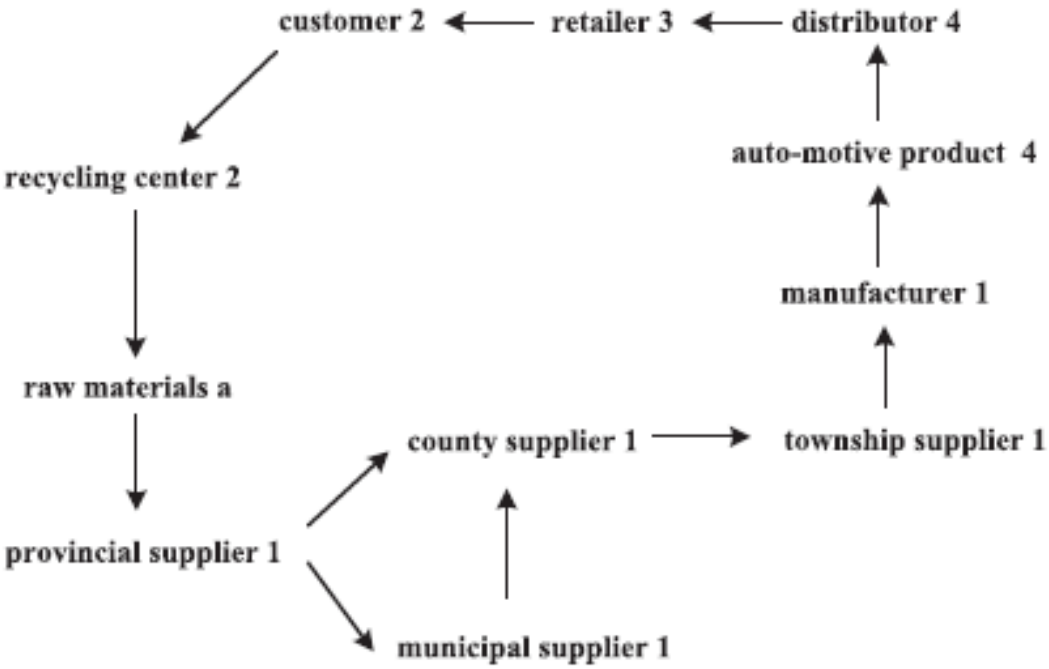


Figure 7. The processing of raw material a
Source: Adapted from [9, Fig. 5]

2. Related Work

The concept of "green supply chain" (GSC) was first introduced at *Michigan State University in the U.S (1996)*. It was the idea that made many manufacturers to reconsider that the process of the traditional supply chain was indeed extremely painful not only for the environmental health but also for their "pockets". This new GSC approach aims to reusable and sustainable materials. Thus, Industrial waste manipulation and recycling centers are the base of the whole idea of "green supply chains". Industrial waste manipulation refers to optimizing waste reduction in Industries. Recycling centers refers to the procedure of recycling useless objects that the customers – users , in other cases, would throw in garbage. The researches cited below extend the topic of green supply chains and are as follows:
The researcher *Benita M .Beamon (1999)* [10] describes the reasons that led to the idea of "green supply chains". More specifically, *Beamon* proposes an extended supply chain which is based on recycling services such as re-manufacturing and reused materials. She highlights the differences between the traditional supply chain and the proposed green supply chain, she discusses the primary reasons and factors that led to the development of a green supply chain in general and presents an alternative approach on evaluating the performance of the proposed supply chain. Lastly, she develops a process for achieving and sustaining the green supply chain. Moreover*, Sameer Kumar , Steve Teichman Tobias Timpernagel (2012)* [11] propose a "green" framework which is characterized by simplicity and it is based on the elimination of waste throughout the supply chain process. The authors believe that if companies rely on it, they can drastically improve their waste management. More specifically, they apply their model to two colossal industries (Coca-Cola, Apple) to prove that this model – framework can make the supply chain more profitable. After all the previous researches and theories, the need for illustration - modelling of a "green supply chain network" became more and more important. Thus, *Mi Pan and Weimin Wu (2014)* [12] using the almighty power of General Stochastic Petri Nets (GSPN), they managed to model a general green supply chain network which was open to subsequent improvement and optimizations. The two researchers also used other tools (e.g., PIPE2) to analyze important factors of the proposed Petri net model such as time performance, economic performance and manufacturing throughput. They concluded that a "green supply chain" improves significantly the previously mentioned factors compared to the traditional supply chain. *L. M. Guevara Ortega, L. Rodríguez Urrego, D. Gómez Gutiérrez and J. G. Chenet (2015)* [13] also moved in the same direction as *Mi Pan, Weimin Wu* did. They selected and evaluated proper mathematical models among others that were used in supply chain processes. After the assessment, they ended up choosing a mathematical model used in green procurement processes based on Colored Petri Nets. They agreed that the proposed model is an excellent solution and meets the requirements of a green procurement process. They also concluded that if the proposed model was applied in practice, it would also satisfy the end-user even more. Last but not least, *Blanka Tundys (2018)* [14] conducted empirical studies and organized a set of tools, statistical and quantitative methods and she used them as the basis for the construction of green supply chains. More specifically, she proposed a model for assessing green supply chains and identified the elements that are used to assess supply chains in Poland. The tools, methods and instruments that *Blanka Tundys* researched can be used to build models, based on mathematics, statistics and operations research, as well as for a higher quality (stochastic) approach and research on the supply chain.

3. Methodology

3.1 Expressions

The notions and ideas described analytically in this Section will be used next, in the description of our CPN Optimized Model. Thus, it is critical to understand them.

3.1.1 Colored Tokens

The main feature of Colored Petri Nets, as described in Section 1.2, is that the tokens are colored. In contrast to traditional Petri Net Models, in which a token is an arbitrary entity that moves dynamically from one location to another, Colored Petri Net Models are made up of tokens that are determined (by color) and have specific attributes. Colored tokens are assigned to specific types of entities that are real-world components of a system, such as red tokens for manufacturers. Furthermore, the timestamp is the primary attribute of a colored token. The timestamp refers to the time when a colored token was created in the system as well as the time delay required for transitions to fire. While we think about time, there is always a universal clock that keeps track of the time. Time advances in time units rather than in real-time measures in a Colored Petri Net (hours, minutes, seconds, etc.). A colored token is defined formally as follows:

Color : Type = <attributes>

Consequently, a colored token has a type that also has attributes. The token attribute can be any data structure that contains stored values. The attributes' values change dynamically. The model designer is responsible for determining which attributes each system entity will have and which values will be affected. Consider the CPN model of Figure 8.

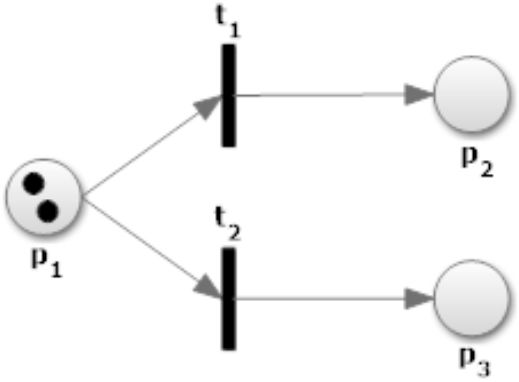


Figure 8. A simple CPN Model

Places
p_1: Vehicles ready
p_2: Ben drives Vehicle
p_3: Jack drives Vehicle

Transitions
t_1: Ben picks vehicle
t_2: Jack picks vehicle

We can add some context to the black tokens of the model by defining the color:

***Black**: Vehicle = <id, weight, timestamp, condition, driver>*. We can symbolize tokens as T_{ID} in graph for convenience. So, we have two black tokens T_1,T_2 that are generated at random time, they carry a timestamp and other attributes. More specifically, the two black tokens can be initialized as it is shown in Figure 9. For example, the black token with ID = 1 is a vehicle that weights 200 units, it was generated at time 2 in place p1, it is new and it is free meaning that it can be driven.

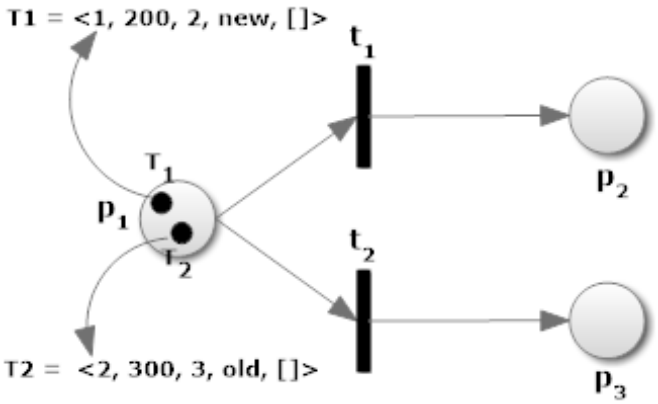


Figure 9. Two initialized discrete black tokens in place p_1

3.1.2 Multifunctional Transitions

Now we'll look at the second extension of Colored Petri Nets in which the transitions are not timeless like in the case of conventional Petri nets but they provide more functionality to precisely describe the temporal details of a system. Transition functionalities include:

1) Firing time. To determine the firing time of a transition, we first examine all of the input locations and then find the token with the shortest time stamp from each location (that is, the next token to leave this place). In the event of a draw between timestamps existing in the same place, the transition algorithm selects the smaller timestamp. After the transition is fired, the time delay is appended to the timestamps of the colored tokens consumed by the transition. When a transition has X input places, we consider X tokens, one token with the shortest time stamp from each of these places. The firing time is then equal to the

maximum of the time stamps chosen, with $t_{i\ max}$ indicating that the last condition required to enable transition t_i became true at time $t_{i\ max}$.

2) Variables that affect a colored token's attribute values. In the CPN Model of Figure 9, for example, a driver name could be stored in the brackets of token attribute <driver> after a transition fires.

3) Other firing rules (Guards) that will be presented in Section 3.1.3

For example, in the case of CPN Model of Figure 9, which transition will fire first and which token will consume?
Firstly, we examine the input places of t_1 and t_2 which is p_1. The token T_1 has timestamp = 2 and token T_2 timestamp = 3:

$t_{1min} = \min(2, 3) = 2 = T_1$
$t_{2min} = \min(2, 3) = 2 = T_1$

The transitions t_1, t_2 are in crash because of the fact that they need T_1 to fire. If there is no other firing rule (guards) rather than firing time, one of the two transitions will randomly fire first. Let's assume that the transition t_1 fires first. After t_1 fires we can define variables that affect the attribute vales of T_1 such as:
T_1.timestamp += 3 (random number for time delay of t_1), T_1.driver.add("Ben").So T_1 will be updated as T_1 = <1, 200, 2 + 3, new, ["Ben"]> in place p_2 as it shown in Figure 10.

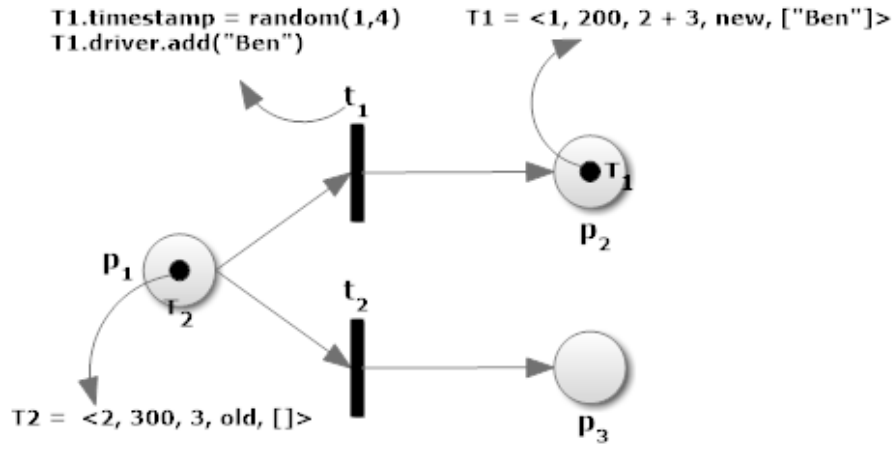


Figure 10. CPN Model status after t_1 fires

3.1.3 Guards

Sometimes, multiple transitions can be enabled at the same time in some cases like the CPN Model of Figure 8 where transitions t1, t2 were in crash. In this case we decided that randomly one of two transitions will fire (t1). This should be avoided, because priority of transition firing renders the model more accurate. In order to provide priority for

transition firing, there must be a kind of referee in the model who decides which transition to fire first. These referees are known as guards or guardian expressions. They are placed exactly below the transitions and are written in parenthesis. Guards provide firing rules for the system to operate more accurate. They are typically control structures that direct the system based on the conditions that are true or false. Thus, we can extend the functionality of the CPN Model in Figure 10 by adding guards in order to prevent crashes. Let it be:

G_1 : t_1 fires (if T_N.weight < 300)
G_2: t_2 fires (if T_N.weight ≥ 300)

Now it is clear that in our example, t_2 cannot fire at timestamp = 2 because the weight of the vehicle of token T_1 is does not meet the requirements. In other words, conditions generate Boolean expressions. If the condition is true, the transition will fire, otherwise, it will not and the system will find in time the transition that meets the requirements of guardians. Furthermore, there can be multiple guardian expressions in a single transition in order to make the model more accurate and less susceptible to crashes. Visually, Figure 11 shows the modification of the CPN Model example.

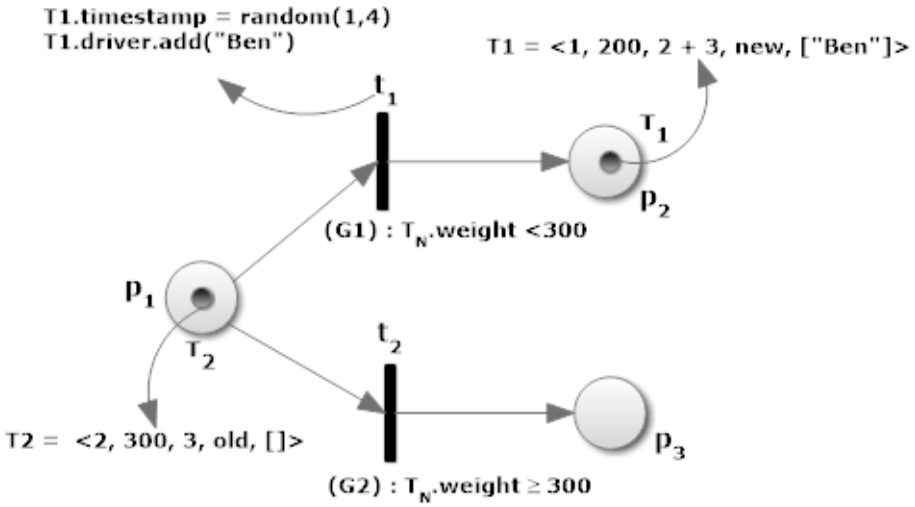


Figure 11. Addition of guards in the CPN Model example

3.1.4 Places

The places in a CPN Model express the temporary condition of the model. In addition, multiple-colored tokens can be placed on them but with the following difference in relation with the conventional Petri nets:
While in conventional Petri nets the places should be fully descriptive because the tokens do not carry information, instead in colored Petri nets places are less descriptive and more generalized. In other words, places describe a general entity and the details of this entity are fully described by the token attributes. For example, if there are four *retailer* entities like our proposed model, we can compress them into one place called *retailers*. The place *retailers* contain colored tokens that are carrying

information for each retailer e.g., variables like retailer1 are stored in a data structure like a list which represents the history of the supply chain. In the example of Figure 4, we do not describe p_1 in details (Vehicle with ID = 1 weights 200 units and it is brand new), instead we do describe p_1 in a more abstract way (Vehicles) and all the details are included in a Vehicle token which is part of category Vehicles that is placed in p_1.

3.2 Disadvantages of the Proposed Model - Advantages of the CPN Optimized Model

The goal of the designers of the proposed model was not so much to present the model of Figure 6 but to use it and to apply on it other methods in order to highlight the hierarchical structure in a green supply chain system. They achieve this with mathematical algorithms such as (PEPA Models) in order to extract the concurrent processes that occur for each material separately. Consequently, the system has several disadvantages which will be presented below.

First of all, there are many redundant places in the proposed Petri net model. For example, the four different types of township suppliers, manufacturers, automobiles, distributors, retailers, customers, recycling centers are presented as four different places. This is not aesthetically pleasing and is not easy for anyone to study it. In addition, the places and transitions of the proposed model are not described in detail and this renders the model more complicated. More specifically, there are cases where a transition has two input positions and two output positions. For example, transition t_{20} describes two concurrent actions:

Automotive product 2 and 3 are sent to distributors 2 and 3.

Although, this creates confusion for the reader because he cannot understand what is the correspondence of an automotive product with a distributor. In the optimized version of the proposed model which will be presented in Section 3.3, the extension of the colored Petri nets is applied to the conventional proposed model. The contributions of our work are the following:

1. Adding colored tokens, the model is able to generate data.

2. Places are decreased. For example, the four different types of manufacturers that were modeled as four different places instead collapse into one place. This is achieved by adding four tokens in a place where each token carries attributes of the different types of manufacturers.

3. The model becomes more realistic. A characteristic of colored Petri nets is that they introduce the concept of time. The time in combination with data carried by the tokens, they are all token attributes. For instance, supply chain history, supply chain costs etc. give the model more functionality and content. Extracting important statistical information with simulations, gives us useful conclusions such as which of the four types of manufacturers offers the most profit to the automotive industry.

3.3 The CPN Optimized Model

This is the main Section of the thesis, in which the CPN Optimized Model will be described thoroughly. The Colored Petri Net Optimized Model is a new version of the proposed Green Supply Chain Petri Net Model in an Automotive Industry.

3.3.1 Overview

The CPN Optimized model is represented visually in Figure 12. It is created with the SmartDraw tool which is a Web-Based Platform for creating diagrams, graphs and other business visuals. The CPN Optimized Model consists of entities such as Suppliers(Provincial, Municipal, County, Township), Manufacturers, Automobiles, Distributors, Retailers, Customers, Recycling Centers. The interaction of these entities is translated into the model in 21 places, 38 transitions, 28 guards and 3 types of colored tokens.

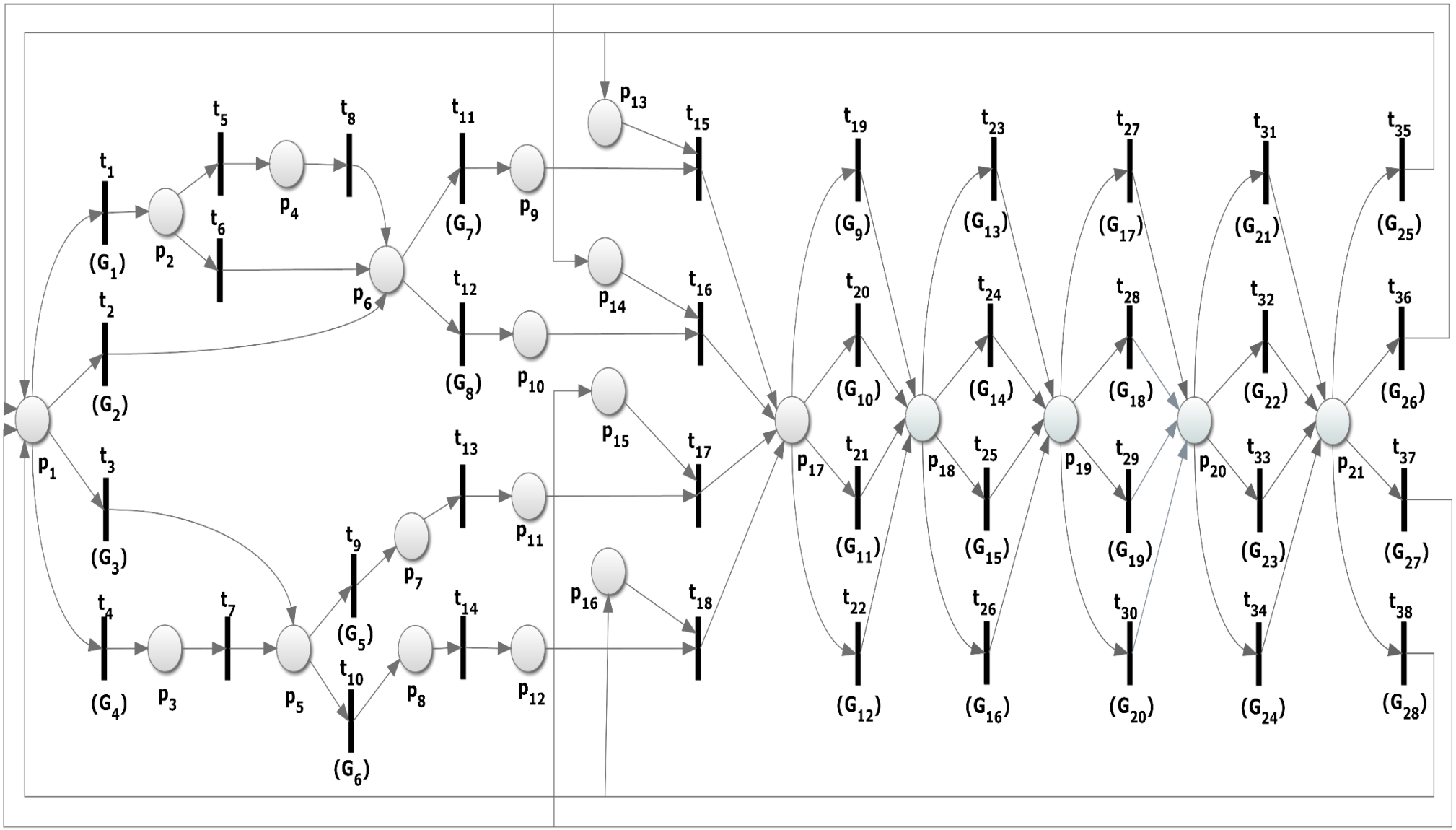


Figure 12. The CPN Optimized Model

Colored Tokens

The colored tokens of the CPN Optimized model are divided into 3 categories:

1) Black tokens which represent the Material entity

2) Red tokens which represent the Manufacturer entity

3) Black-Red tokens which represent the produced automobile entity.

Speaking in a more object-oriented way, we could say that Black-Red tokens inherit both Black and Red tokens. (Multiple Inheritance).
The colored tokens are defined as follows:

Black : Material = <material_id, material_type, history, cost, timestamp, material_weight> (place p_1)

Attributes – Values:

1) **material_id** → positive integer starting from 1.Each token that is created has its one unique ID

2) **material_type = "a" or "b" or "c" or "d"** → a string that indicates the type of material. The attribute is initialized in (p_1) between four types of strings , "a", "b", "c", "d"

3) **history = [ ]** → a list which stores information (strings) about material life cycle. When a token is created it is initialized as an empty list

4) **cost = random(300, 501)** → indicates a positive integer that rises the value of material every time it changes hands from suppliers to manufacturers. It is initialized as a random number between 150 to 200 units

5) **timestamp = random(1,11)** → indicates the creation time (time units) of the token and it increases every time the token is consumed by a transition. It is initialized as a random number between 1-time unit to 10-time units

6) **material_weight = random(700, 801)** → integer that indicates the quantity of the material it will be supplied and therefore the quantity it will be used to produce an automobile

A black token initialization in p_1 could be:

T_1 = <1, "a", [ ], 160, 5>

Red : Manufacturer = <manufacturer_id, manufacturer_type, income, expenses>
(place p_{13}, p_{14}, p_{15}, p_{16})

Attributes – Values:

1) **manufacturer_id** → positive integer starting from 1.Each token that is created has its one unique ID. If other token IDs have been registered the ID value takes the next positive integer from the max value of the already existing IDs

2) **manufacturer_type = "1" or "2" or "3" or "4"** → a string that indicates the manufacturer_type. In contrast to black tokens where four material_types are randomly initialized in one place (p_1), in the case of red tokens, four places represent each manufacturer_type. In other words, each manufacturer_type has its own unique initialization place and the other attributes have common initialization. The manufacturer_type mappings are:

place p_{13} → Manufacturer.manufacturer_type = "1"
place p_{14} → Manufacturer.manufacturer_type = "2"
place p_{15} → Manufacturer.manufacturer_type = "3"
place p_{16} → Manufacturer.manufacturer_type = "4"

3) **income = 0** → the value amount (positive integer) that the manufacturer gains after selling a automobile. It is initialized as 0

4) **expenses = 0** → the value amount positive integer that manufacturer spends from the moment the automobile produced until is being received from retailer. The expenses also include the material supply costs

We don't use timestamps for red tokens because their timestamps are identical to the waiting time of the black tokens, so it's pointless to define them. As a result, the timestamp carried by the black token becomes the timestamp for the red-black token's creation.
A red token initialization in p_{14} could be:
T_3 = <3, "2", 0, 0>

Black-Red : Automobile = <**Material**, **Manufacturer**, id, automobile_type> (place p_{17})

Attributes – Values:

1) **Material** → It is written in capitals because it refers to black token Material with all its attributes. In other words, a black-red token inherits all black token's attributes from which it was created. Thus, the black-red tokens are superclass of the black tokens

2) **Manufacturer** → It is written in capitals because it refers to red token Manufacturer with all its attributes. In other words, a black-red token inherits all red token's attributes from which it was created. Thus, the black-red tokens are superclass of the red tokens

3) **id** → positive integer.Each token that is created has its one unique ID. If other token IDs have been registered the ID value takes the next positive integer from the max value of the already existing IDs

4) **automobile_type = "1" or "2" or "3" or "4"** → a string that indicates the automobile_type. The attribute is initialized in (p_{17}) between four automobiles types, 1, 2, 3, 4

The transition that creates the black-red token in place p_{17} e.g., t_{15} calculates the variable automobile production cost and adds at the existing Material.cost the automobile production cost

Places

p_1 : Materials
p_2 : Provincial supplier 1
p_3 : Provincial supplier 2
p_4 : Municipal supplier 1
p_5 : Municipal supplier 2
p_6 : County supplier 1
p_7 : County supplier 2
p_8 : County supplier 3
p_9 : Township supplier 1
p_{10} : Township supplier 2
p_{11} : Township supplier 3
p_{12} : Township supplier 4
p_{13} : Manufacturer 1
p_{14} : Manufacturer 2
p_{15} : Manufacturer 3
p_{16} : Manufacturer 4
p_{17} : Automobiles
p_{18} : Distributors
p_{19} : Retailers
p_{20} : Customers
p_{21} : Recycling Centers

Transitions

t_1 : Raw material a is supplied by provincial supplier 1
effects: Material.cost += random(150, 201)
Material.history.add("provincial supplier 1")
Material.timestamp += random(2, 6)
t_2 : Raw material c is supplied by county supplier 1
effects: Material.cost += random(450, 601)
Material.history.add("county supplier 1")
Material.timestamp += random(6, 15)

t_3 : Raw material d is supplied by municipal supplier 2
effects: Material.cost += random(300, 401)
Material.history.add("municipal supplier 2")

Material.timestamp += random(4, 10)

t_4 : Raw material b is supplied by provincial supplier 2
effects: Material.cost += random(150, 201)
Material.history.add("provincial supplier 2")
Material.timestamp += random(2, 6)

t_5 : Raw material a is supplied from provincial supplier 1 to municipal supplier 1
effects: Material.cost += random(150, 201)
Material.history.add("municipal supplier 1")
Material.timestamp += random(2, 6)

t_6 : Raw material a is supplied from provincial supplier 1 to county supplier 1
effects: Material.cost += random(300, 401)
Material.history.add("county supplier 1")
Material.timestamp += random(4, 10)

t_7 : Raw material b is supplied from provincial supplier 2 to municipal supplier 2
effects: Material.cost += random(150, 201)
Material.history.add("municipal supplier 2")
Material.timestamp += random(2, 6)

t_8 : Raw material a is supplied from municipal supplier 1 to county supplier 1
effects: Material.cost += random(150, 201)
Material.history.add("county supplier 1")
Material.timestamp += random(2, 6)

t_9 : Raw material b is supplied from municipal supplier 2 to county supplier 2
effects: Material.cost += random(150, 201)
Material.history.add("county supplier 2")
Material.timestamp += random(2, 6)

t_{10} : Raw material d is supplied from municipal supplier 2 to county supplier 3
effects: Material.cost += random(150, 201)
Material.history.add("county supplier 3")
Material.timestamp += random(2, 6)

t_{11} : Raw material a is supplied from county supplier 1 to township supplier 1
effects: Material.cost += random(150, 201)
Material.history.add("township supplier 1")
Material.timestamp += random(2, 6)

t_{12} : Raw material c is supplied from county supplier 1 to township supplier 2

effects: Material.cost += random(150, 201)
Material.history.add("township supplier 2")
Material.timestamp += random(2, 6)

t_{13} : Raw material b is supplied from county supplier 2 to township supplier 3

effects: Material.cost += random(150, 201)
Material.history.add("township supplier 3")
Material.timestamp += random(2, 6)

t_{14} : Raw material d is supplied from county supplier 3 to township supplier 4

effects: Material.cost += random(150, 201)
Material.history.add("township supplier 4")
Material.timestamp += random(2, 6)

t_{15} : Manufacturer 1 uses raw material a to produce automobile 1

effects: Create Automobile token object which contains Material and Manufacturer token objects (In object-oriented programming, it is a derived class from Material and Manufacturer)
Automobile.history.add("manufacturer 1", "automobile 1")
Automobile.automobile_type = "1"
Automobile.timestamp += random(15, 21)
Automobile.cost += random(2500, 5001)
Automobile.expenses +=
Automobile.cost

t_{16} : Manufacturer 2 uses raw material c to produce automobile 2

effects: Create Automobile token object which contains Material and Manufacturer token objects (In object-oriented programming, it is a derived class from Material and Manufacturer)
Automobile.history.add("manufacturer 2", "automobile 2")
Automobile.automobile_type = "2"
Automobile.timestamp += random(15, 21)
Automobile.cost += random(2500, 5001)
Automobile.expenses +=
Automobile.cost

t_{17} : Manufacturer 3 uses raw material b to produce automobile 3

effects: Create Automobile token object which contains Material and Manufacturer token objects (In object-oriented programming, it is a derived class from Material and Manufacturer)
Automobile.history.add("manufacturer 3", "automobile 3")
Automobile.automobile_type = "3"
Automobile.timestamp += random(15, 21)
Automobile.cost += random(2500, 5001)
Automobile.expenses +=
Automobile.cost

t_{18} : Manufacturer 4 uses raw material d to produce automobile 4

effects: Create Automobile token object which contains Material and

Manufacturer token objects (In object-oriented programming, it is a derived class from Material and Manufacturer)

Automobile.history.add(“manufacturer 4”, “automobile 4”)
Automobile.automobile_type = “4”
Automobile.timestamp += random(15, 21)
Automobile.cost += random(2500, 5001)
Automobile.expenses +=
Automobile.cost

t_{19} : Automobile 1 is sent to distributor 1

effects:

Automobile.expenses += random(150, 201)
Automobile.history.add(“distributor 1”)
Automobile.timestamp += random(3, 7)

t_{20} : Automobile 2 is sent to distributor 2

effects:

Automobile.expenses += random(150, 201)
Automobile.history.add(“distributor 2”)
Automobile.timestamp += random(3, 7)

t_{21} : Automobile 3 is sent to distributor 3

effects:

Automobile.expenses += random(150, 201)
Automobile.history.add(“distributor 3”)
Automobile.timestamp += random(3, 7)

t_{22} : Automobile 4 is sent to distributor 4

effects:

Automobile.expenses += random(150, 201)
Automobile.history.add(“distributor 4”)
Automobile.timestamp += random(3, 7)

t_{23} : Distributor 1 assigns automobile product 1 to retailer 1

effects:

Automobile.expenses += random(150, 201)
Automobile.history.add(“retailer 1”)
Automobile.timestamp += random(3, 7)

t_{24} : Distributor 2 assigns automobile product 2 to retailer 2

effects:

Automobile.expenses += random(150, 201)
Automobile.history.add(“retailer 2”)
Automobile.timestamp += random(3, 7)

t_{25} : Distributor 3 assigns automobile product 3 to retailer 3

effects:

Automobile.expenses += random(150, 201)
Automobile.history.add(“retailer 4”)
Automobile.timestamp += random(3, 7)

t_{26} : Distributor 4 assigns automobile product 4 to retailer 4
effects:
Automobile.expenses += random(150, 201)
Automobile.history.add("retailer 3")
Automobile.timestamp += random(3, 7)

t_{27} : Retailer 1 sells automobile product 1 to customer 1
effects:
Automobile.income += random(7000, 9001)
Automobile.history.add("customer 1")
Automobile.timestamp += random(3, 7)

t_{28} : Retailer 2 sells automobile product 2 to customer 2
effects:
Automobile.income += random(7000, 9001)
Automobile.history.add("customer 2")
Automobile.timestamp += random(3, 7)

t_{29} : Retailer 3 sells automobile product 3 to customer 3
effects:
Automobile.income += random(7000, 9001)
Automobile.history.add("customer 3")
Automobile.timestamp += random(3, 7)

t_{30} : Retailer 4 sells automobile product 4 to customer 4
effects:
Automobile.income += random(7000, 9001)
Automobile.history.add("customer 4")
Automobile.timestamp += random(3, 7)

t_{31} : Customer 1 returns automobile product 1 to recycling center 1
effects:
Automobile.history.add("recycling center 1")
Automobile.timestamp += random(2, 6)

t_{32} : Customer 2 returns automobile product 2 to recycling center 2
effects:
Automobile.history.add("recycling center 2")
Automobile.timestamp += random(2, 6)

t_{33} : Customer 3 returns automobile product 3 to recycling center 3
effects:
Automobile.history.add("recycling center 3")
Automobile.timestamp += random(2, 6)

t_{34} : Customer 4 returns automobile product 4 to recycling center 4
effects:
Automobile.history.add("recycling center 4")
Automobile.timestamp += random(2, 6)

t_{35} : Recycling Center 1 sorts, recycles automobile 1
effects:
Automobile.history.add("recycled")
Automobile.timestamp += random(10, 16)
Initialize a red token in place p_{13}
Manufacturer.manufacturer_type = "1"
Initialize a black token in place p_1
(Material.material_type = "a" and Material.cost -= 0,5 Material.cost)

t_{36} : Recycling Center 2 sorts, recycles automobile 2
effects:
Automobile.history.add("recycled")
Automobile.timestamp += random(10, 16)
Initialize a red token in place p_{14}
Manufacturer.manufacturer_type = "2"
Initialize a black token in place p_1
(Material.material_type = "c" and Material.cost -= 0,5 Material.cost)

t_{37} : Recycling Center 3 sorts, recycles automobile 3
effects:
Automobile.history.add("recycled")
Automobile.timestamp += random(10, 16)
Initialize a red token in place p_{15}
Manufacturer.manufacturer_type = "3"
Initialize a black token in place p_1
(Material.material_type = "b" and Material.cost -= 0,5 Material.cost)

t_{38} : Recycling Center 4 sorts, recycles automobile 4
effects:
Automobile.history.add("recycled")
Automobile.timestamp += random(10, 16)
Initialize a red token in place p_{16}
Manufacturer.manufacturer_type = "4"
Initialize a black token in place p_1
(Material.material_type = "d" and Material.cost -= 0,5 Material.cost)

Guards

(G_1) : t_1 fires (if Material.material_type = "a")

(G_2) : t_2 fires (if Material.material_type = "c")

(G_3) : t_3 fires (if Material.material_type = "d")

(G_4) : t_4 fires (if Material.material_type = "b")

(G_5) : t_9 fires (if Material.material_type = “b”)
(G_6) : t_{10} fires (if Material.material_type = “d”)

(G_7) : t_{11} fires (if Material.material_type = “a”)

(G_8) : t_{12} fires (if Material.material_type = “c”)

(G_9) : t_{19} fires (if Automobile.automobile_type = “1”)

(G_{10}) : t_{20} fires (if Automobile.automobile_type = “2”)

(G_{11}) : t_{21} fires (if Automobile.automobile_type = “3”)

(G_{12}) : t_{22} fires (if Automobile.automobile_type = “4”)

(G_{13}) : t_{23} fires (if “distributor 1” in Automobile.history)

(G_{14}) : t_{24} fires (if “distributor 2” in Automobile.history)

(G_{15}) : t_{25} fires (if “distributor 3” in Automobile.history)

(G_{16}) : t_{26} fires (if “distributor 4” in Automobile.history)

(G_{17}) : t_{27} fires (if “retailer 1” in Automobile.history)

(G_{18}) : t_{28} fires (if “retailer 2” in Automobile.history)

(G_{19}) : t_{29} fires (if “retailer 3” in Automobile.history)

(G_{20}) : t_{30} fires (if “retailer 4” in Automobile.history)

(G_{21}) : t_{31} fires (if “customer 1” in Automobile.history)

(G_{22}) : t_{32} fires (if “customer 2” in Automobile.history)

(G_{23}) : t_{33} fires (if “customer 3” in Automobile.history)

(G_{24}) : t_{34} fires (if “customer 4” in Automobile.history)

(G_{25}) : t_{35} fires (if “recycling center 1” in Automobile.history)

(G_{26}) : t_{36} fires (if “recycling center 2” in Automobile.history)

(G_{27}) : t_{37} fires (if “recycling center 3” in Automobile.history)

(G_{28}) : t_{38} fires (if “recycling center 4” in Automobile.history)

3.3.2 Initial Markings

As stated in the introduction, the initial markings are critical because they determine the model's correct operation. If, for example, the model's

designer did not create it properly so that it works perfectly, serious problems such as deadlocks can occur, necessitating a redesign of the model. In this case, as we are the designers of the CPN Optimized Model, we define the following general initial marking structures:
-four types of **Red** tokens based on automobile_type attribute:
T_N = <manufacturer_id, "1", income, expenses> (place p_{13})
T_N = <manufacturer_id, "2", income, expenses> (place p_{14})
T_N = <manufacturer_id, "3", income, expenses> (place p_{15})
T_N = <manufacturer_id, "4", income, expenses> (place p_{16})
Because the red tokens do not carry a timestamp we suppose that are created in 0 time units.
-four types of **Black** tokens based on material_type attribute:
T_N = <material_id, "a", history, cost, timestamp, material_weight> (place p_1)
T_N = <material_id, "b", history, cost, timestamp, material_weight > (place p_1)
T_N = <material_id, "c", history, cost, timestamp, material_weight > (place p_1)
T_N = <material_id, "d", history, cost, timestamp, material_weight > (place p_1)

Let's assume now that we have a generator that initializes the attribute values:
(The specific and random values of attributes are mentioned in Section 3.3.1)

Red tokens:
T_1 = <1, "1", 0, 0> (place p_{13})
T_2 = <2, "2", 0, 0> (place p_{14})
T_3 = <3, "3", 0, 0> (place p_{15})
T_4 = <4, "4", 0, 0> (place p_{16})
Black tokens:
T_5 = <5, "b", [ ], 320, 1, 730> (place p_1)
T_6 = <6, "c", [ ], 300, 3, 750> (place p_1)
T_7 = <7, "a", [ ], 400, 6, 780> (place p_1)
T_8 = <8, "d", [ ], 350, 8, 785> (place p_1)

The Figure 13 highlights visually the initial marking:

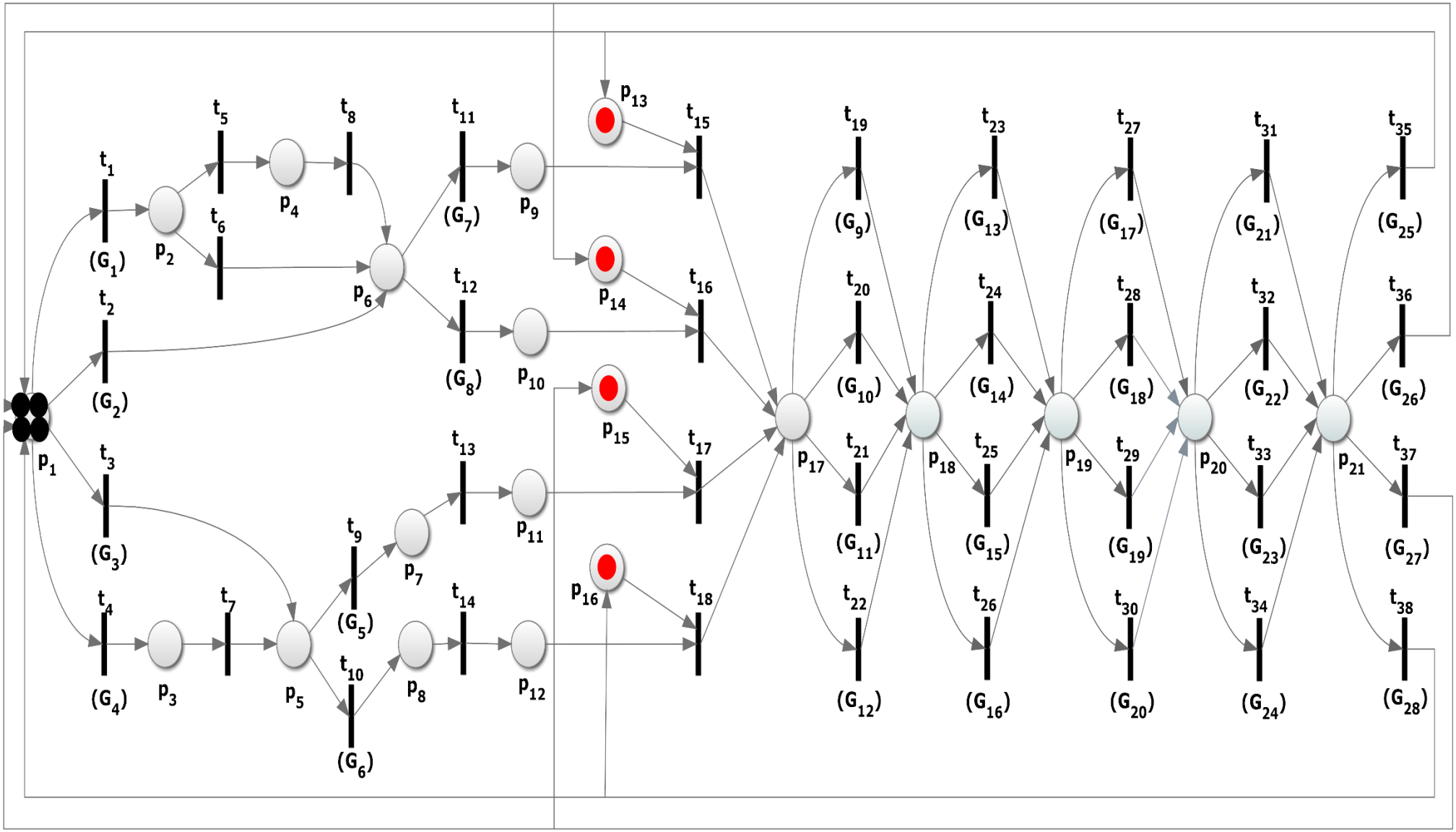


Figure 13. Initial Markings of the CPN Optimized Model

3.3.3 Reachability Graph – Deadlocks

As mentioned in the introduction, if a marking occurs with no transition activated, the network will become deadlocked. This means that within a Petri net, there are numerous undesirable situations. The detection of these situations is of utmost importance for the implementation of a model with Petri nets. The answer to the question "which sequences of network firings lead to a deadlock;" is the subject of the so-called reachability problem. The reachability tree or reachability graph is the analysis method used to answer the above question. The reachability tree consists of a tree with Petri net-labeled nodes (ancestors - descendants) and transition arcs that represent every possible firing of all enabled transitions. The states of the Petri net model are stored in the reachability tree nodes (which are represented as rectangles visually). The Petri net's states change dynamically as the tree's nodes spread. The nodes are expressions that describe all of the model's discrete states. The states are defined as a set of places and the corresponding tokens for each place. Each place is described as $\mathbf{P_N^{(i)}}$, where **N** is the place's id and **i** is the number of tokens contained in the place. If a descendant node is a subset of or identical to the initial marking, i.e., it consists of the same places with fewer or equal tokens, then we can say that the model is deadlocked free. In case we have multiple descendants, all descendants must be identical or subsets of the initial marking. Now we will approve that the CPN Optimized Model is deadlocked free. Figure 14 shows the Reachability-Graph of the CPN Optimized Model. The tree comes to an end at descendant nodes that are colored red. This means that all terminal nodes are a subset of the original marking.

Figure 14. Reachability-Graph of the CPN Optimized Model

3.3.4 Execution Example

In this Section, we will show an execution example of the CPN Optimized Model. Particularly, we will isolate and deal with a black token that has material_type "a" and we will show its movement in the system until a **Black-Red** token (automobile) is created. For better understanding, we will consider each transition firing in the system as a step. Let's assume that the tokens we are interested in is defined by a generator as follows:
T_7 = <7, "a", [ ], 400, 6, 730> (place p_1) (**Black**)
T_1 = <1, "1", 0, 0> (place p_{13}) (**Red**)

Step 1
t_1 fires because G_1 is true.
effects: T_7 = <7, "a", ["provincial supplier 1"], 560, 9, 730> (place p_2)

Step 2
t_5 or t_6 fires (No Guards)
Let's assume that t_6 fires randomly.
effects: T_7 = <7, "a", ["provincial supplier 1", "county supplier 1"], 910, 11, 730> (place p_6)

Step 3
t_{11} fires because of G_7.
effects: T_7 = <7, "a", ["provincial supplier 1", "county supplier 1", "township supplier 1"], 1080, 15, 730> (place p_9)

Step 4
t_{15} fires (No Guards) and manufacturer 1 creates automobile 1. Thus, a black-red token is created.
effects: T_9 = <7 (material_id), "a" (material_type), "1" (manufacturer_type), 1 (manufacturer_id), ["provincial supplier 1", "county supplier 1", "township supplier 1", "manufacturer 1", "automobile 1"] (history), 730 (material_weight), 16 (timestamp), 0 (income), 1080 (expenses), 9 (id), "1" (automobile_type)> (place p_9).

Similarly, the green supply chain cycle continues until the type 1 automobile is recycled, at which point a new cycle begins, with the cost of the material "a" reduced because a portion of it has been recycled from type 1 automobile. Below, in Figures 15, 16, 17, 18 we can see the 4 steps of executing the mode.

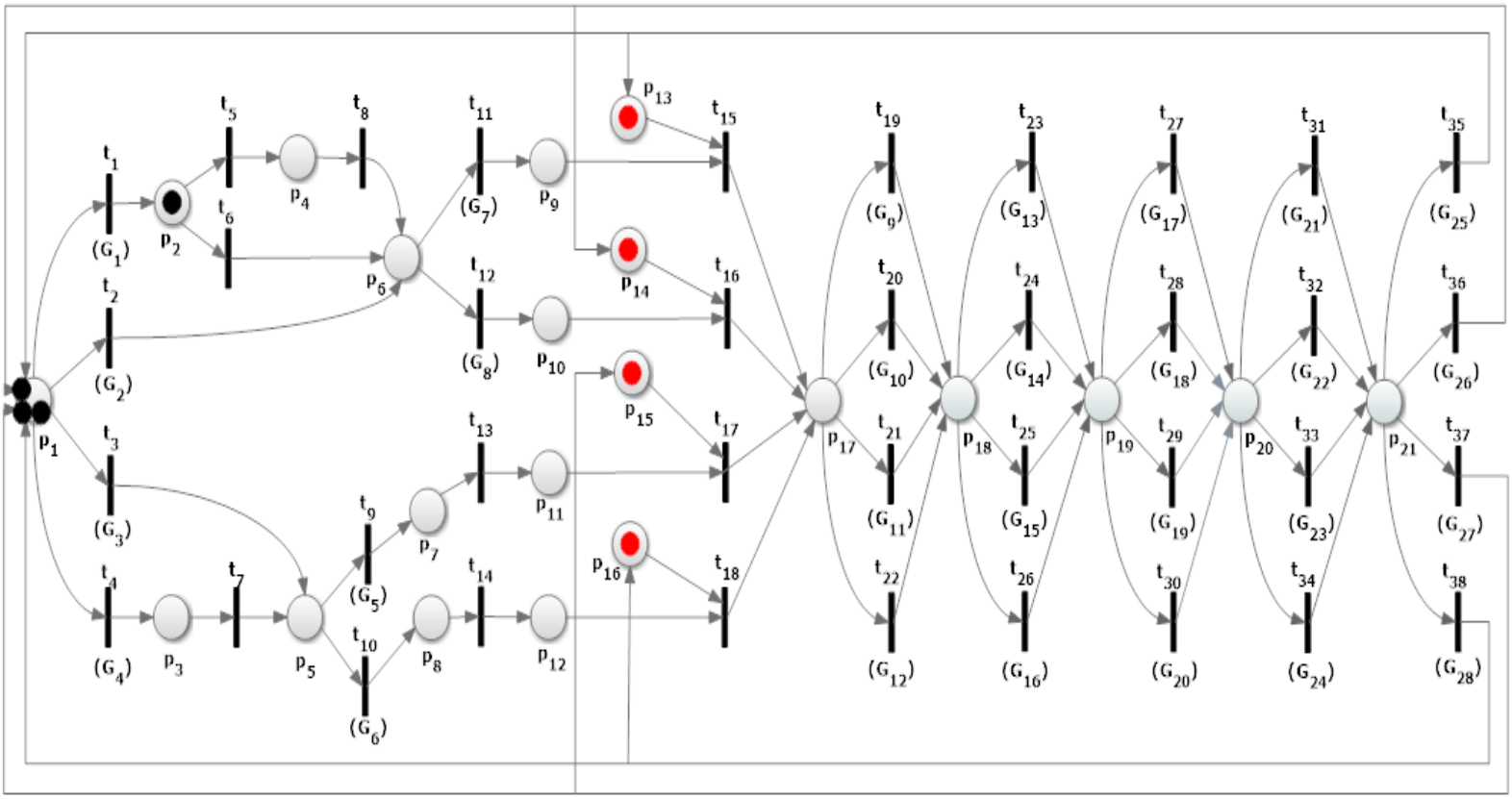


Figure 16. Execution Example - Step 1

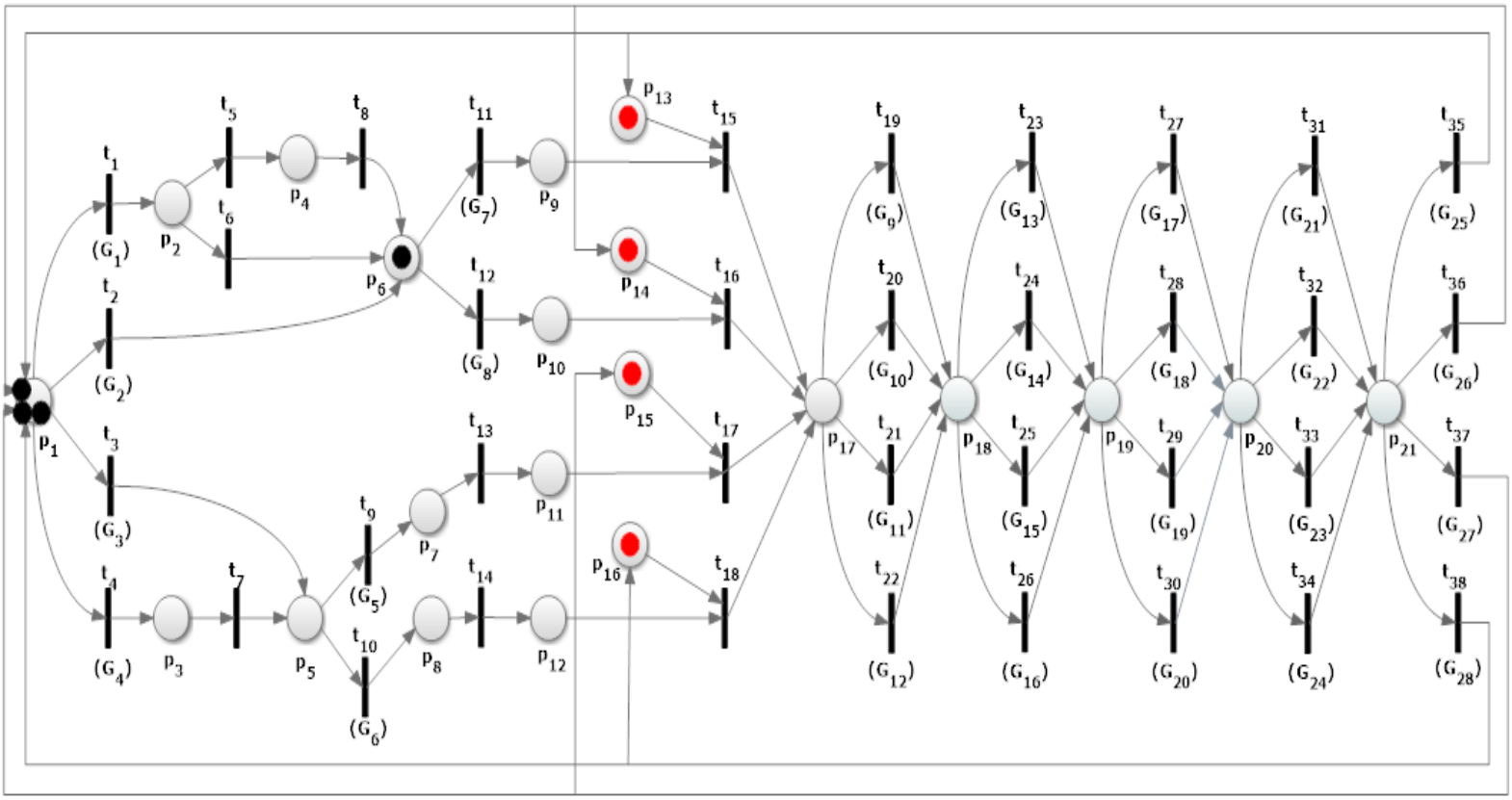


Figure 15. Execution Example - Step 2

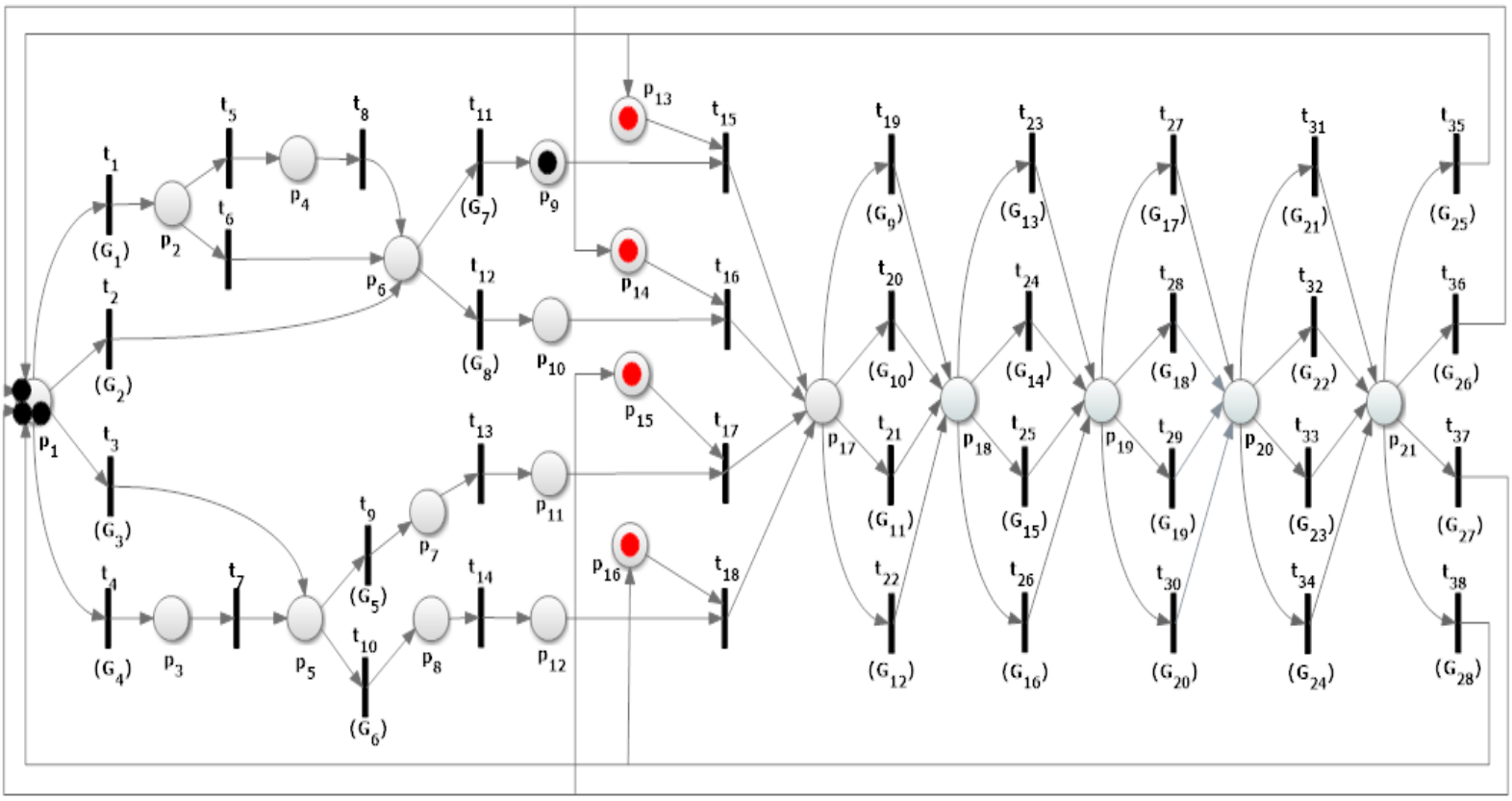


Figure 17. Execution Example - Step 3

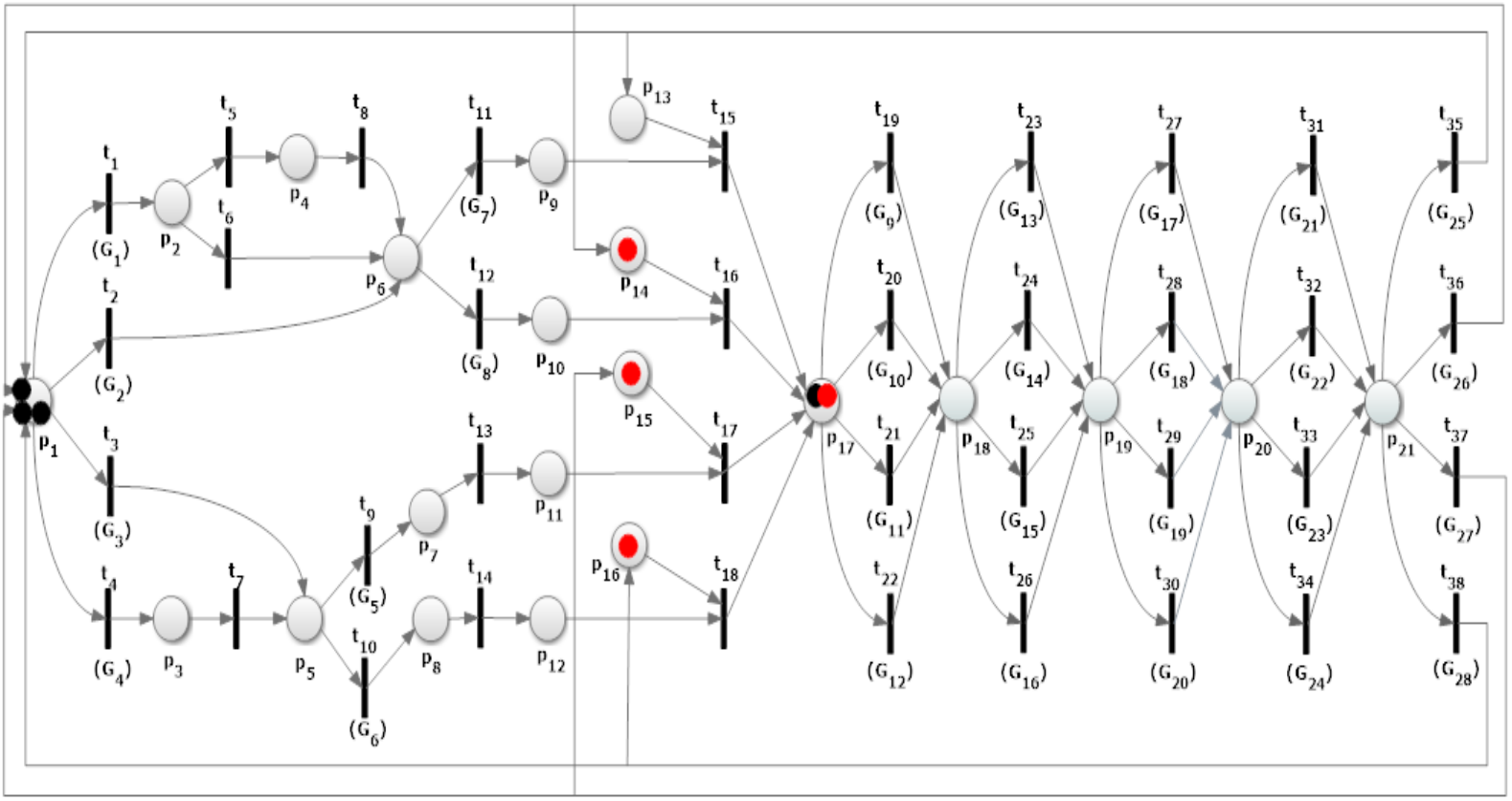


Figure 18. Execution Example - Step 4

4. Programming of the CPN Optimized Model in Python

Running a model and extracting information from it on paper is a time-consuming and laborious process. As a result, we require the assistance of a program to implement the model and directly extract results from the number of simulations we enter to the program. With the term simulation, we mean the representation of our model's operation. This means that one simulation, for example, is equivalent to one execution representation of our model. The implemented program is based on object-oriented programming. Its function is as follows:

1) it prints the system's dynamic process to the console (such as moving tokens and influencing its variables)
2) it saves the data generated for each automobile's construction in .txt files
3) it outputs statistics based on the number of simulations entered by the user

The mapping of the system entities with their representation in the program is:

- **Colored Tokens → Class Objects**
  Each colored token in the system has a unique id and its attributes – variables that are affected.Thus, in python we can simulate colored tokens as class instances and their attributes as class members. Moreover, class methods can be added to classes to provide more functionality (such as the get_profits() method of this Class Automobile). The program consists of three Classes:

  1) Material (Black Tokens)
  2) Manufacturer (Red Tokens)
  3) Automobile (which is a derived class from both Material and Manufacturer and represents the Black-Red Tokens)

  The Figures 19, 20, 21 highlight the programming of the three classes.

```
class Material:
    def __init__(self, material_id, material_type):
        self.material_id = material_id
        self.material_type = material_type
        self.history = []
        self.cost = randrange(300, 501)
        self.timestamp = randrange(1, 11)
        self.material_weight = randrange(700, 801)
```

Figure 19. Class Material

```
class Manufacturer:
    def __init__(self, manufacturer_id, manufacturer_type):
        self.manufacturer_id = manufacturer_id
        self.manufacturer_type = manufacturer_type
        self.income = 0
        self.expenses = 0
```

Figure 20. Class Manufacturer

```
from material import Material
from manufacturer import Manufacturer


class Automobile(Material, Manufacturer):
    def __init__(self, material_id, material_type, manufacturer_id, manufacturer_type, history, timestamp,
                 income, expenses, id, automobile_type):
        Material.__init__(self, material_id, material_type)
        Manufacturer.__init__(self, manufacturer_id, manufacturer_type)
        self.history = history
        self.timestamp = timestamp
        self.income = income
        self.expenses = expenses
        self.id = id
        self.automobile_type = automobile_type
```

Figure 21. Class Automobile

- **Places → Lists**
  In order to add tokens in a place, there must be a data structure that can store any tokens inside it. For this reason, the lists are a very good representation of the positions of the system.

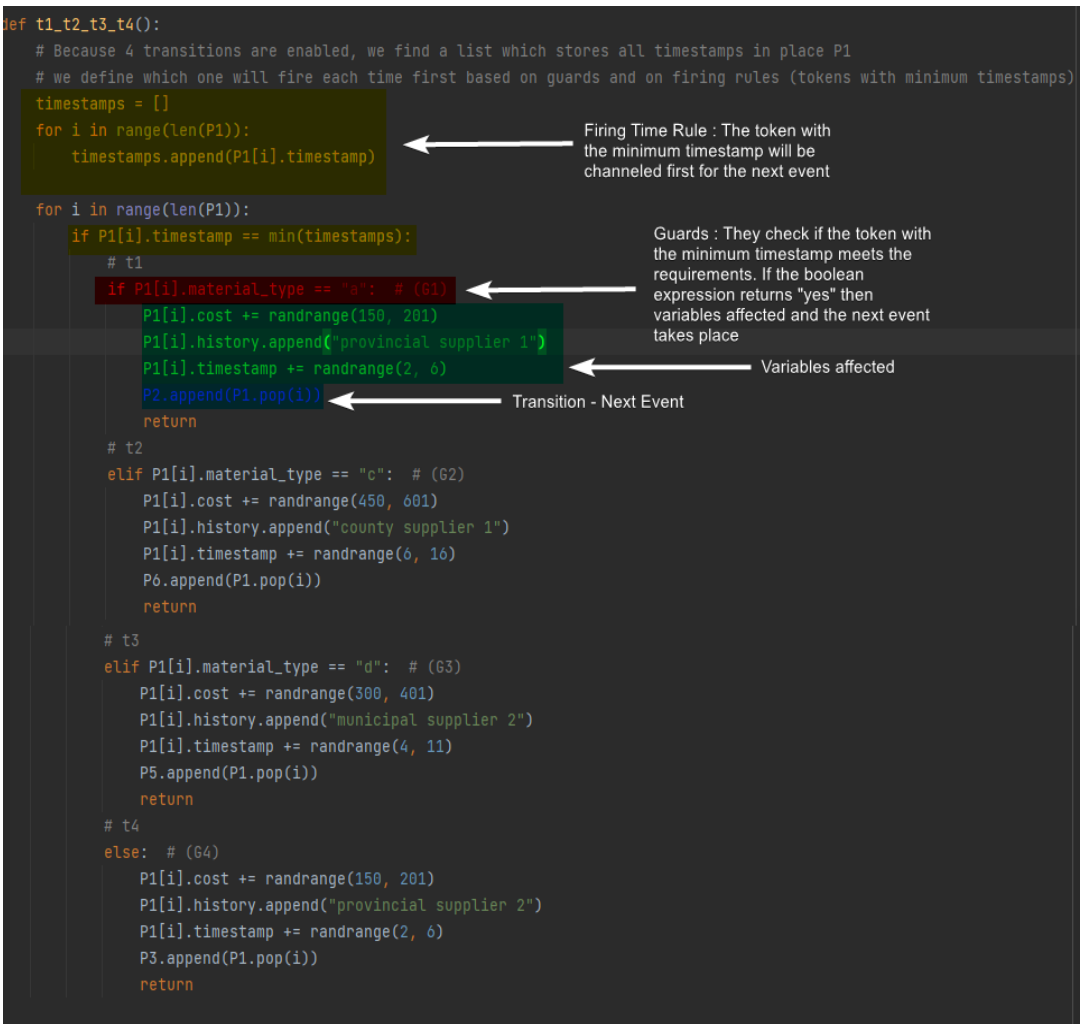
```
from material import Material
from manufacturer import Manufacturer
from automobile import Automobile
from random import randrange, random, uniform

# Globals - Places
P1 = []
P2 = []
P3 = []
P4 = []
P5 = []
P6 = []
P7 = []
P8 = []
P9 = []
P10 = []
P11 = []
P12 = []
P13 = []
P14 = []
P15 = []
P16 = []
P17 = []
P18 = []
P19 = []
P20 = []
P21 = []
```

Figure 22. Places - Lists

- **Transitions – Guards → Functions/Methods**
  As described in Section 3.1.2, 3.1.3, the transitions render the system dynamic affecting variables by directing tokens from place to place , creating new instances of a Class (such as the initial marking), imposing firing rules(guards, firing times). Consequently, functions/methods are one way of representing transitions. Inside the scope of a function are integrated all the other functionalities of a transition (variables, guards, class instances). In order to prevent creating a function for every transition in the system, we can collapse a branching of more than two transitions with one input place into one function. The Figure 23 highlights the function that integrates the branching of four

transitions t_1, t_2, t_3, t_4 of the input place P_1.




```
def t1_t2_t3_t4():
    # Because 4 transitions are enabled, we find a list which stores all timestamps in place P1
    # we define which one will fire each time first based on guards and on firing rules (tokens with minimum timestamps)
    timestamps = []
    for i in range(len(P1)):
        timestamps.append(P1[i].timestamp)

    for i in range(len(P1)):
        if P1[i].timestamp == min(timestamps):
            # t1
            if P1[i].material_type == "a":  # (G1)
                P1[i].cost += randrange(150, 201)
                P1[i].history.append("provincial supplier 1")
                P1[i].timestamp += randrange(2, 6)
                P2.append(P1.pop(i))
                return
            # t2
            elif P1[i].material_type == "c":  # (G2)
                P1[i].cost += randrange(450, 601)
                P1[i].history.append("county supplier 1")
                P1[i].timestamp += randrange(6, 16)
                P6.append(P1.pop(i))
                return
            # t3
            elif P1[i].material_type == "d":  # (G3)
                P1[i].cost += randrange(300, 401)
                P1[i].history.append("municipal supplier 2")
                P1[i].timestamp += randrange(4, 11)
                P5.append(P1.pop(i))
                return
            # t4
            else:  # (G4)
                P1[i].cost += randrange(150, 201)
                P1[i].history.append("provincial supplier 2")
                P1[i].timestamp += randrange(2, 6)
                P3.append(P1.pop(i))
                return
```

Figure 23. Function t1_t2_t3_t4()

- **Colored Petri Net → (If - While) Structures**
  The Figure 24 highlights the coding of the Colored Petri Net Model with If-While statements. Every time a place has x tokens (if), there is a (while) condition that runs until it makes sure that the tokens have left the specific place and have been consumed by the corresponding functions-transitions (or branching transitions) according to the corresponding firing rules.

```
for i in range(simulations):
    print("")
    print(f"{40 * '~'} Simulation {i + 1} {210 * '~'}\n")
    print_petri_net()
    if P1:
        while len(P1) >= 1:
            t1_t2_t3_t4()
            print_petri_net()
    if P2:
        while len(P2) >= 1:
            t5_t6()
            print_petri_net()
    if P3:
        while len(P3) >= 1:
            t7()
            print_petri_net()
    if P4:
        while len(P4) >= 1:
            t8()
            print_petri_net()
    if P5:
        while len(P5) >= 1:
            t9_t10()
            print_petri_net()
```

Figure 24. Colored Petri Net Coding

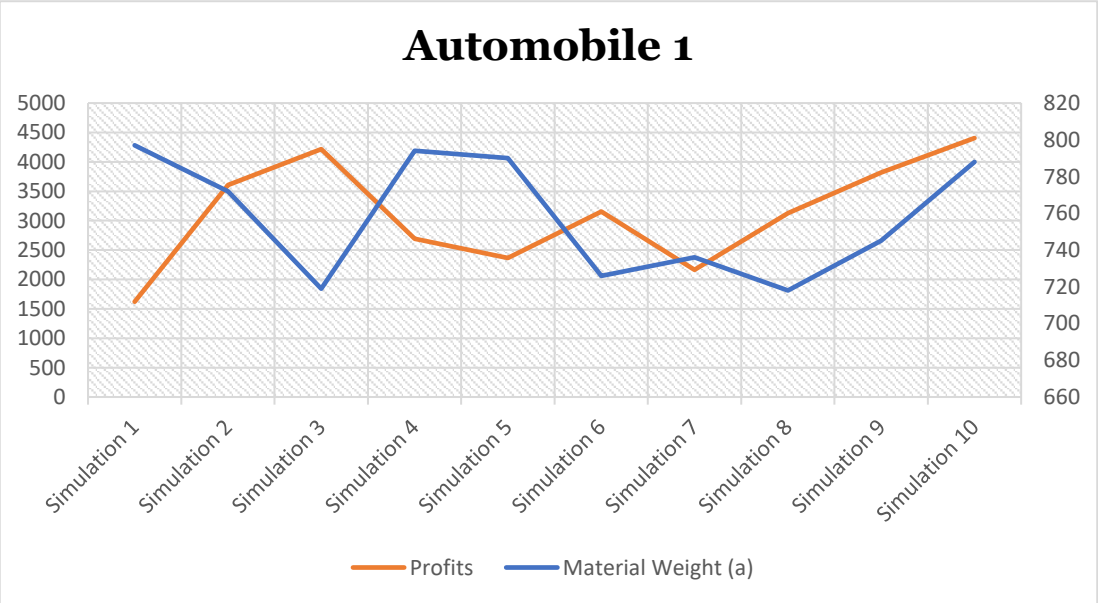
Although the program is quite extensive, we have highlighted only its key features. For more details, the code is available at:

https://github.com/apostolisSrg/Thesis-Implementation

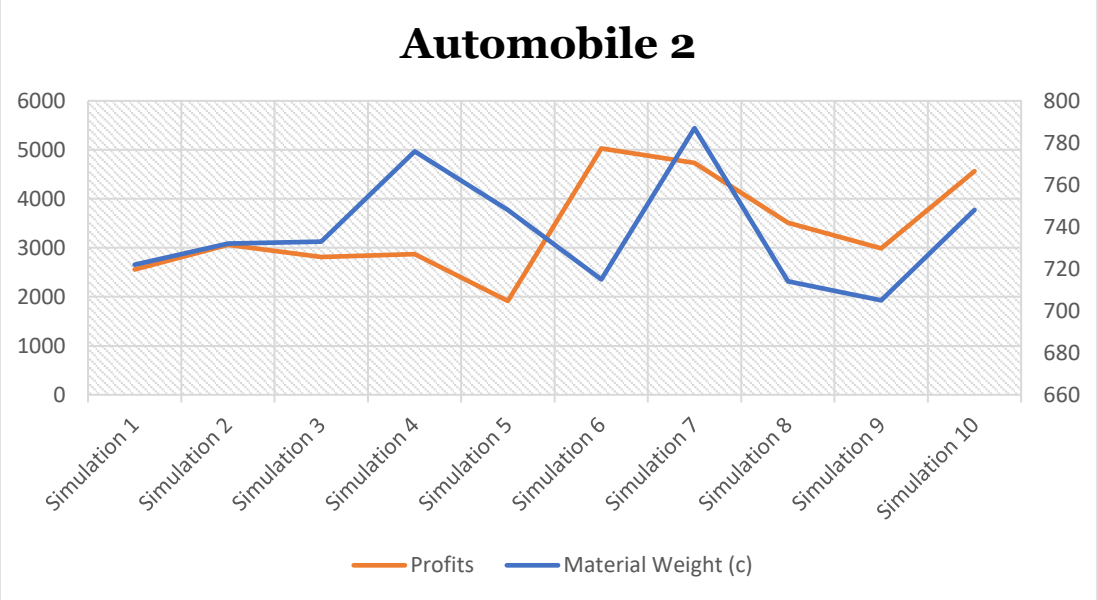
5. Simulation Results

In this Section, we will draw charts based on the number of simulations we would like to run in order to derive a conclusion about the efficiency of manufacturers and automobiles. First of all, if someone examines the code referenced in Section 4, they will see that the program asks for the number of simulations to run. After that, if the user inputs the number of simulations, the program saves the outcome data in .txt files for each simulation and automobile. Thus, we will use these data to combine information in order to have a clearer picture of the efficiency, utilization of each automobile. The simulation results are random, so there is not something standard to draw a general conclusion. If the attributes – variables of the program are not random numbers but they are exact data that reflect reallity or they get values based on a specific algorithm, then data outcome is crucial for those who want to make a specific optimization (e.g., to increase the profits of automobile 1). Moreover, the attribute values (e.g., expenses) are not the real values but they represent the real values (e.g., expenses = 100 money units). Consequently, this can be adapted to everyone's need in real-time. Although, this thesis focuses on demonstrating the simulation results in a more abstract way. The program ran 10 Simulations and the results are presented below:

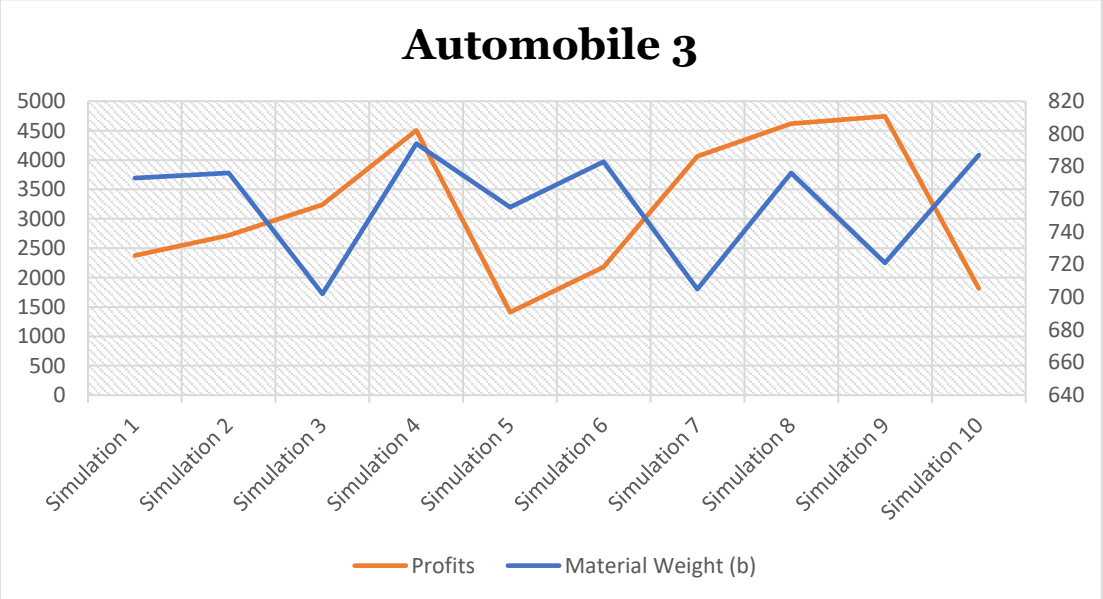
Profits – Material Weight



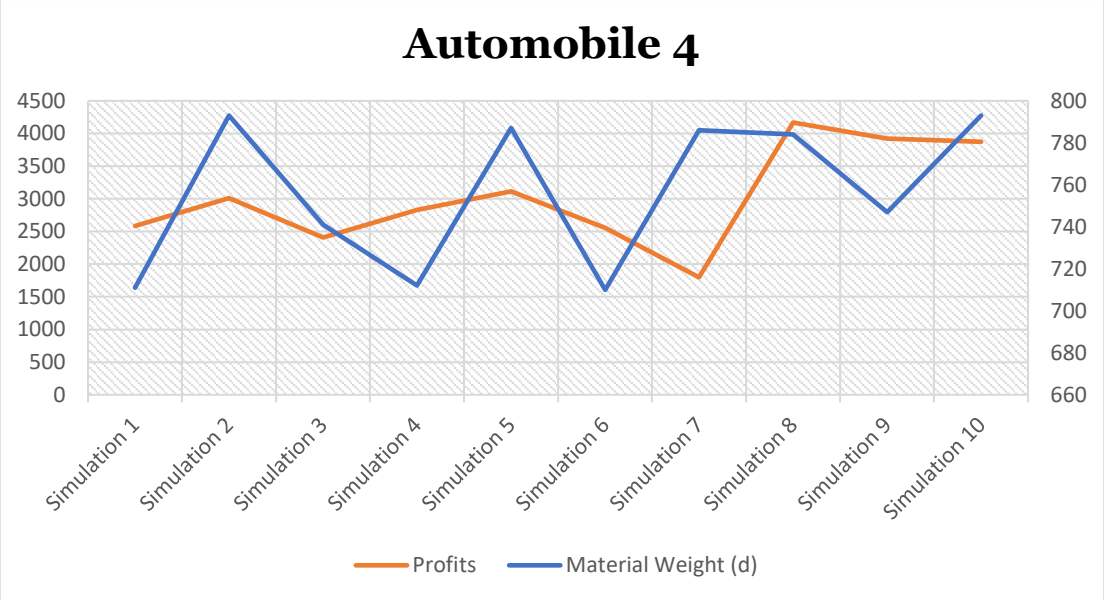
Comments : Manufacturer 1 seems to work more efficiently in simulations 1 to 3 and 5 because it manages to obtain simultaneously more profits and to reduce the weight of the material required for the construction of the Automobile 1.



Comments : Manufacturer 2 seems to work more efficiently in simulation 6. In the remaining simulations either the profits drop down and the material weight inflates or profits and material weight have proportional fluctuation.

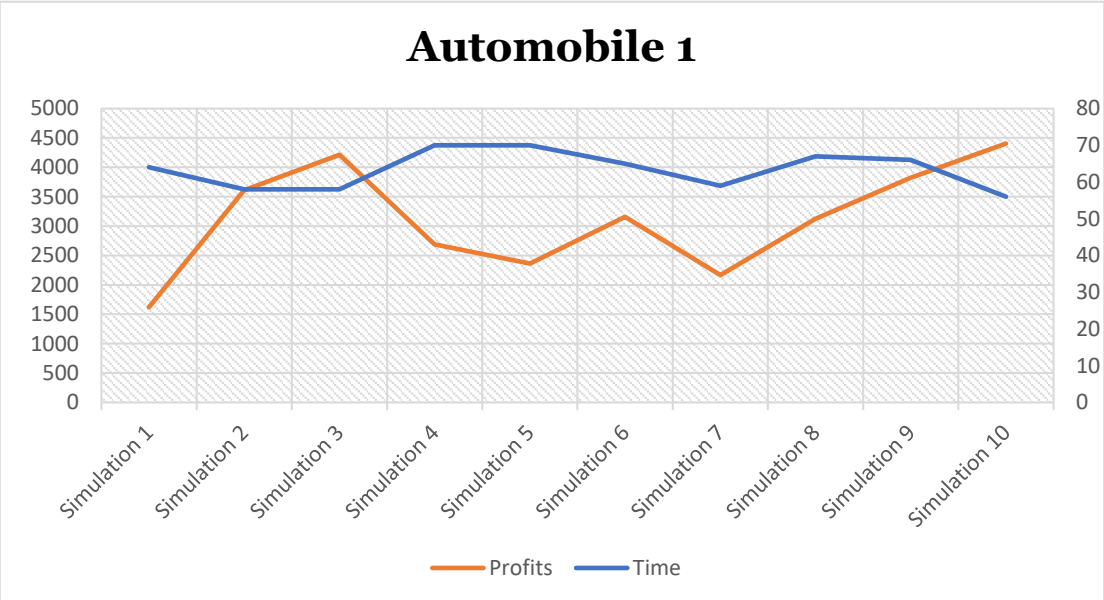


Comments : Manufacturer 3 seems to work more efficiently in simulations 3 and 7. In the remaining simulations either the profits drop down and the material weight inflates or profits and material weight have proportional fluctuation.

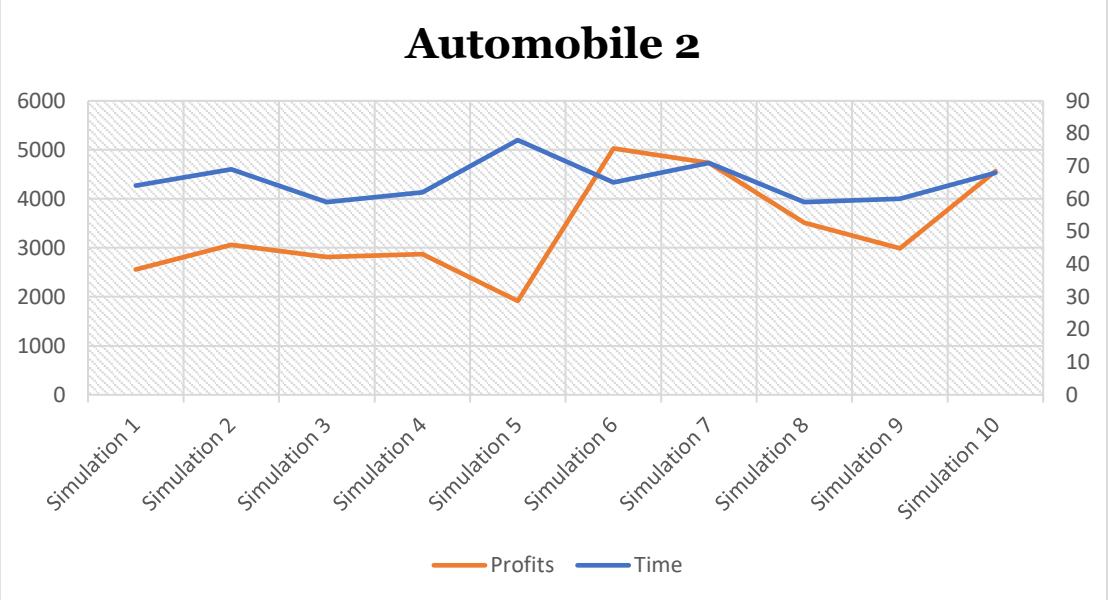


Comments : Manufacturer 4 seems to work more efficiently in simulation 4. Usually, when the material weight d increases, the Automobile's 4 profits are less than the previous simulations.

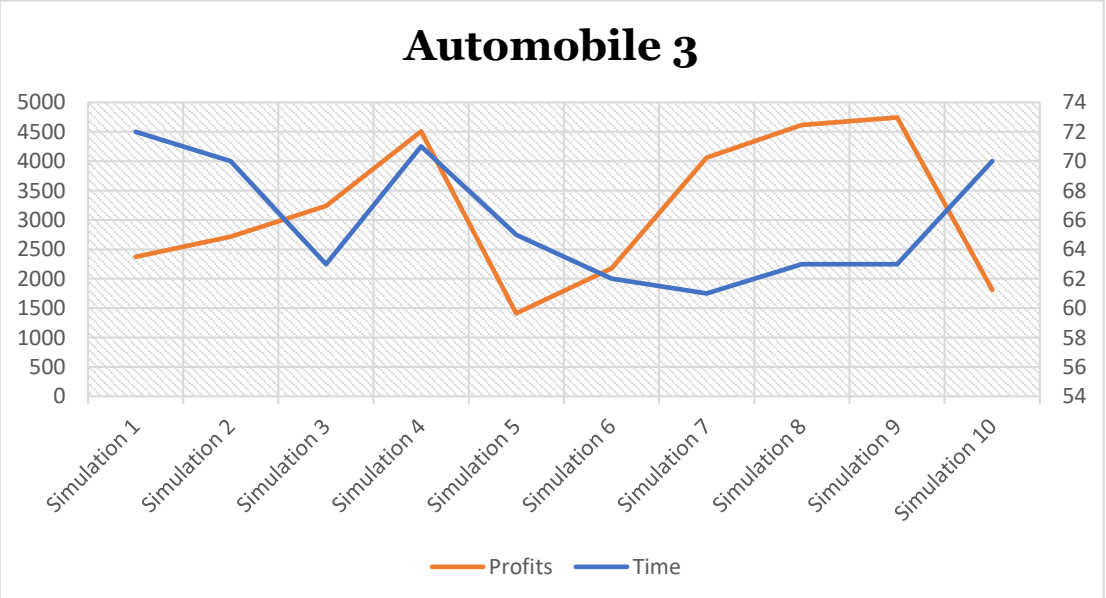
Profits – Time



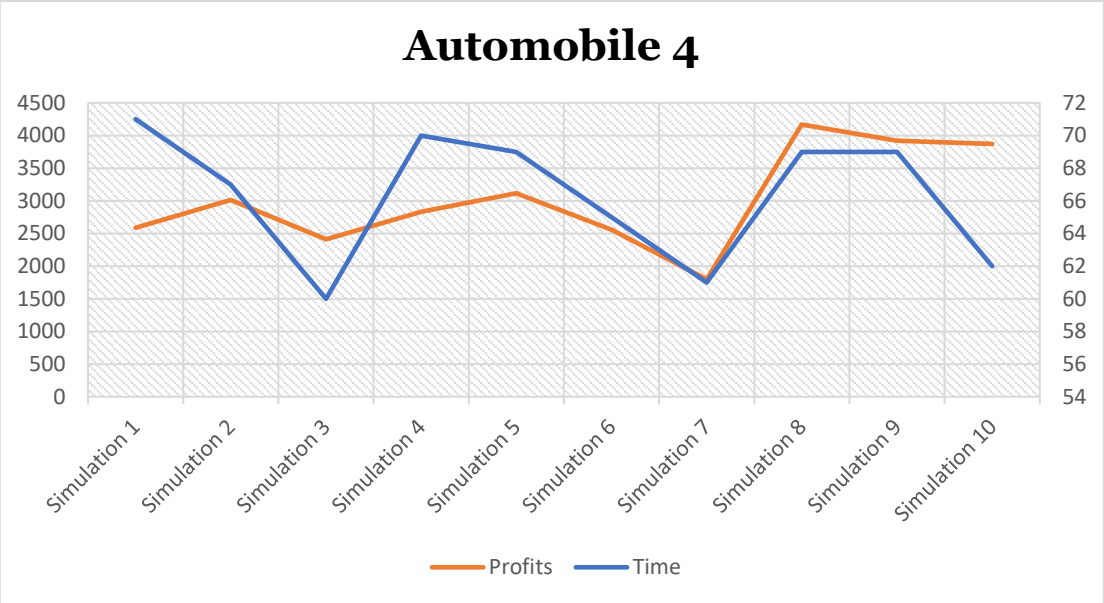
Comments : Manufacturer 1 produces faster and yields more in simulations 1 to 3. This process comes to its peak at simulation 10 where Manufacturer 1 reaps the the most profits (4404 money units) in the least amount of time (56 time units).



Comments : Manufacturer 2 has a sharp decrease in profits and a simultaneous increase in the production time of Automobile 2 in the simulation 5. After that, in simulation 6 the reverse occurs and the manufacturer reaps the most profits (5029)

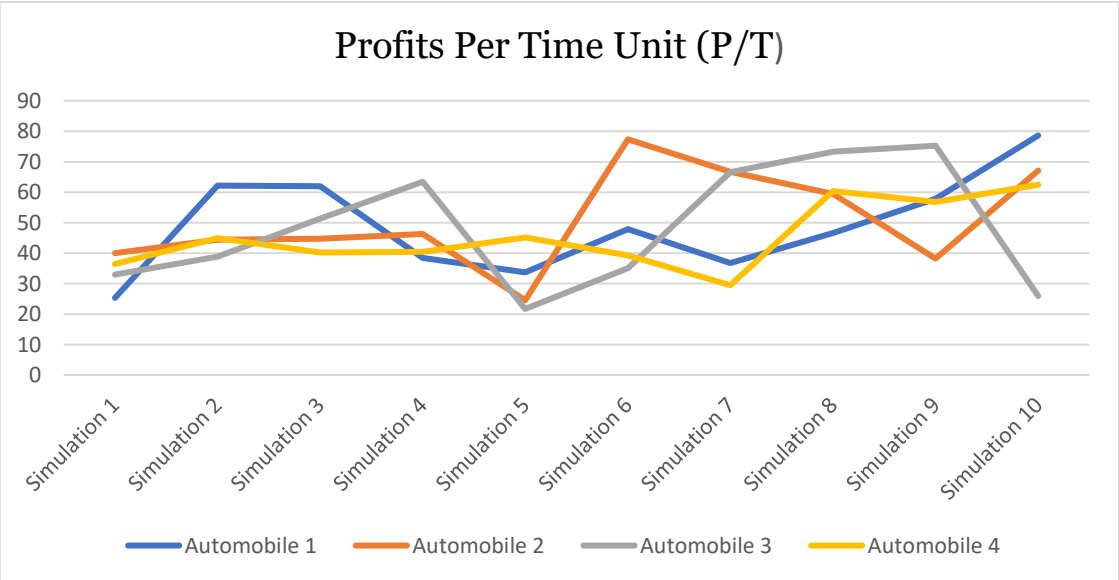


Comments : Manufacturer 3 has a slight increase in profits and a simultaneous decrease in produce time in simulations 1 to 3. Crucial is the fact that in simulation 10 there is a steep decrease in profits and a steep incraese in produce time. That renders the simulation 10 a failed industry year for Manufacturer 3.



Comments : Manufacturer 4 approximately reaps profits in analogy to time. In other words, the fluctuation of the product time causes the fluctuation to profits. That means, the more time Manufacturer 4 needs to produce Automobile 4 the more profits he will earn.

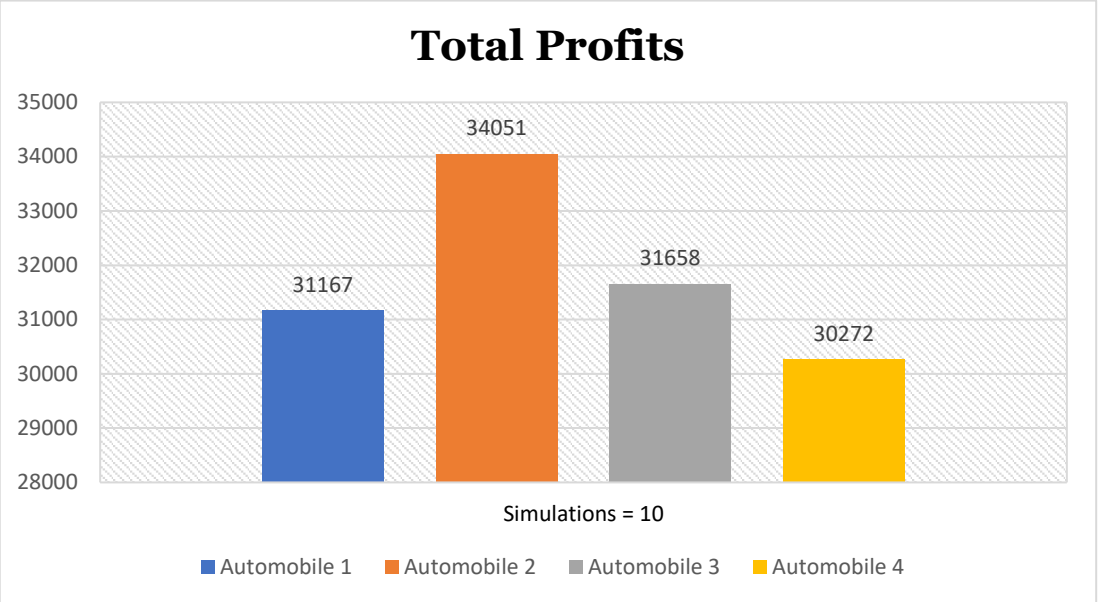
Profits Per Time Unit (P/T)



Comments : If we compare the profit made per unit of time by each manufacturer in each simulation, we see that there are significant differences in some points. For example, in simulations 3 and 4, Manufacturer 1 (who

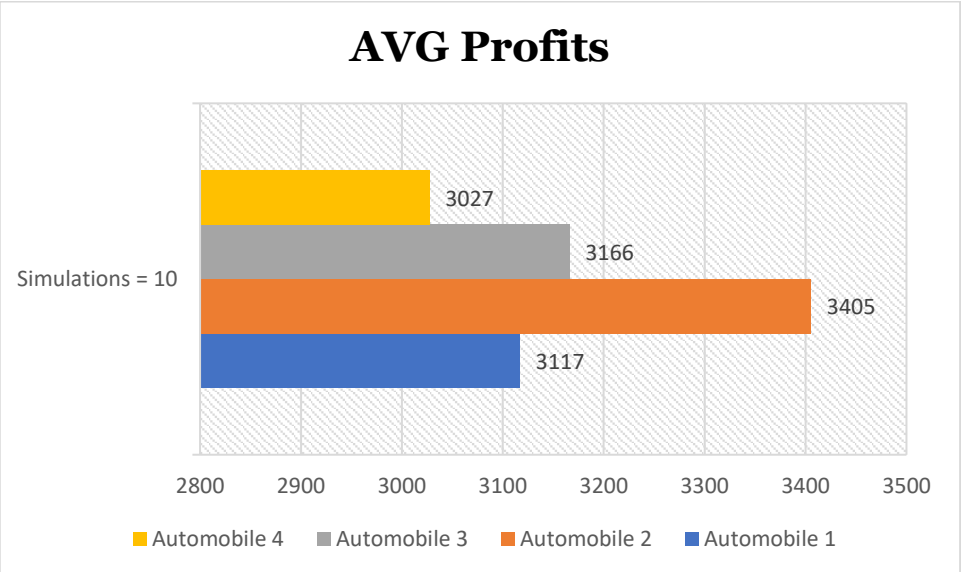
constructs Automobile 1) is more productive and profitable. Also, Manufacturer 2 has a steep increase in his productivity and profitability in simulation 6.

Total Profits



Comments : Manufacturer 2 reaped the most profits in the industry overall. This means that the Automobile 2 turned out to be more successful.

AVG Profits



Comments : The Automobile 2 has an average profit of 3405

Conclusion - Future Work

To summarize, the goal of this thesis was to highlight colored Petri nets (a variation on traditional Petri nets) in order to improve an already existing and simpler model. The improved model was presented and thoroughly examined. It was investigated whether there were any deadlocks using reachability graphs, and some examples of running the model were provided. The model was then programmed in Python so that it could be run and we could see the results. Thus, we ran ten simulations in order to reach some statistical conclusions. The program's goal was not to optimize data output (for example, using an optimization algorithm), but to represent the process of green supply chains in the automotive industry. Suggestions for future improvements to the model include the following:

- **Further improvement of the model**
  The materials used to construct each car should be more specific and less abstract (like type a material). Furthermore, the entities could be more specific, such as what type an automobile is (luxury or everyday). Another possibility is to have production sectors. A manufacturing sector, for example, may produce more than one automobile. Finally, additional agents that affect the model's variables differently, such as cannibalization, refurbishing, direct reuse, and disposal, could be added to the recycling centers. All these suggestions contribute to a more detailed model that would be more accurate and closer to real world application in automotive industries. Moreover, it could be more flexible for applying optimization algorithms

- **Variable values are assigned based on a specific type**
  (e.g., car production cost = material.weight * material.cost * timestamp)

- **Application of optimization algorithms** (e.g., how to increase the total income with the least cost)

Bibliography

[1] Sadegh Ekrami, “Bindings in Colored Petri Nets,” Thesis (Masters), Department of Computer Science Memorial University of Newfoundland, St. John’s, Canada, 2014. [Online]. Available: https://research.library.mun.ca/8272

[2] Μάνος Ρουμελιώτης and Σταύρος Σουραβλάς, *ΤΕΧΝΙΚΕΣ ΠΡΟΣΟΜΟΙΩΣΗΣ - ΘΕΩΡΙΑ ΚΑΙ ΕΦΑΡΜΟΓΕΣ*, 1st ed. ΤΖΙΟΛΑ, 2012.

[3] L. Ngalamou and L. Myers, “Petri Nets and Fuzzy Sets in Hybrid Controllers Synthesis: The Discrete-Event Aspect,” vol. 4, no. 2, p. 22, 2009.

[4] “Coloured Petri net,” *Wikipedia*. Aug. 28, 2021. Accessed: Oct. 06, 2022. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Coloured_Petri_net&oldid=1041112706

[5] G. S. Hijjawi, “Impact of Green Supply Chain on Supply Chain Performance,” *WSEAS TRANSACTIONS ON BUSINESS AND ECONOMICS*, vol. 19, pp. 442–452, Jan. 2022, doi: 10.37394/23207.2022.19.40.

[6] E. Manavalan, M. Thanigai Arasu, and J. Kandasamy, “Chapter 14 - Sustainable supply chain management in manufacturing industries,” in *Sustainable Manufacturing*, K. Gupta and K. Salonitis, Eds. Elsevier, 2021, pp. 367–389. doi: 10.1016/B978-0-12-818115-7.00010-9.

[7] Johnny C. Ho, Columbus, Maurice K, Shalishali, Tzu-Liang, and David S. Ang, “Opportunities in GSCM,” *Coastal Business Journal*, vol. 8, no. 1, 2009.

[8] R. Saada, *Green Transportation in Green Supply Chain Management*. IntechOpen, 2020. doi: 10.5772/intechopen.93113.

[9] J. Ding, X. Chen, H. Sun, W. Yan, and H. Fang, “Hierarchical structure of a green supply chain,” *Computers & Industrial Engineering*, vol. 157, p. 107303, Jul. 2021, doi: 10.1016/j.cie.2021.107303.

[10] B. M. Beamon, “Designing the green supply chain,” *Logistics Information Management*, vol. 12, no. 4, pp. 332–342, Aug. 1999, doi: 10.1108/09576059910284159.

[11] S. Kumar, S. Teichman, and T. Timpernagel, “A green supply chain is a requirement for profitability,” *International Journal of Production Research*, vol. 50, no. 5, pp. 1278–1296, Mar. 2012, doi: 10.1080/00207543.2011.571924.

[12] M. Pan and W. Wu, “A Petri Net Approach for Green Supply Chain Network Modeling and Performance Analysis,” in *Business Process Management Workshops*, vol. 171, N. Lohmann, M. Song, and P. Wohed, Eds. Cham: Springer International Publishing, 2014, pp. 330–341. doi: 10.1007/978-3-319-06257-0_26.

[13] L. M. G. Ortega, L. R. Urrego, D. G. Gutiérrez, and J. G. Chenet, “Green procurement model using Petri nets: a perspective developed from the models applied to the supply chain,” presented at the ENERGY AND SUSTAINABILITY 2015, Medellin, Colombia, Sep. 2015, pp. 267–277. doi: 10.2495/ESUS150231.

[14] B. Tundys, “Use of Quantitative and Qualitative Methods for Modelling Green Supply Chains,” *OSCM: An Int. Journal*, pp. 82–97, Mar. 2018, doi: 10.31387/oscm0310205.

[15] N. Viswanadham and N. R. Srinivasa Raghavan, “Performance analysis and design of supply chains: a Petri net approach,” *Journal of the Operational*

Research Society, vol. 51, no. 10, pp. 1158–1169, Oct. 2000, doi: 10.1057/palgrave.jors.2600063.

[16] D. S. Saydam, "An Overview of Green Supply Chain, Green Supply Chain implementation and Organizational Factors," vol. 7, no. 8, p. 15, 2015.

[17] J. Liu and R. Wu, "A Petri Net-based Supply Chain System," *Int. J. Onl. Eng.*, vol. 14, no. 11, p. 28, Nov. 2018, doi: 10.3991/ijoe.v14i11.9502.

[18] Q. Zhu, J. Sarkis, and K. Lai, "Choosing the right approach to green your supply chains," *MSCRA*, vol. 1, no. 1, pp. 54–67, Feb. 2019, doi: 10.1108/MSCRA-02-2019-0006.

[19] Liang Tang, Li Hui, Liang Xu, and Ke Jing, "An Object-oriented Petri Net for supply chain system analysis based on networked manufacturing," in *Proceedings 2011 International Conference on Transportation, Mechanical, and Electrical Engineering (TMEE)*, ChangChun, China, Dec. 2011, pp. 200–203. doi: 10.1109/TMEE.2011.6199179.

[20] M. Dong and F. F. Chen, "Process modeling and analysis of manufacturing supply chain networks using object-oriented Petri nets," *Robotics and Computer-Integrated Manufacturing*, vol. 17, no. 1–2, pp. 121–129, Feb. 2001, doi: 10.1016/S0736-5845(00)00045-4.

[21] M. Bevilacqua, F. E. Ciarapica, and G. Mazzuto, "Modelling and performance analysis of a supply chain using timed coloured Petri nets," *IJBPSCM*, vol. 4, no. 3/4, p. 285, 2012, doi: 10.1504/IJBPSCM.2012.050386.

[22] G. Mazzuto, M. Bevilacqua, and F. E. Ciarapica, "Supply chain modelling and managing, using timed coloured Petri nets: a case study," *International Journal of Production Research*, vol. 50, no. 16, pp. 4718–4733, Aug. 2012, doi: 10.1080/00207543.2011.639397.

[23] L. Liu, X. Liu, and G. Liu, "The risk management of perishable supply chain based on coloured Petri Net modeling," *Information Processing in Agriculture*, vol. 5, no. 1, pp. 47–59, Mar. 2018, doi: 10.1016/j.inpa.2017.12.001.

[24] H. Pierreval, R. Bruniaux, and C. Caux, "A continuous simulation approach for supply chains in the automotive industry," *Simulation Modelling Practice and Theory*, vol. 15, no. 2, pp. 185–198, Feb. 2007, doi: 10.1016/j.simpat.2006.09.019.

[25] S. Kumar, S. Teichman, and T. Timpernagel, "A green supply chain is a requirement for profitability," *International Journal of Production Research*, vol. 50, no. 5, pp. 1278–1296, Mar. 2012, doi: 10.1080/00207543.2011.571924.

[26] A. Ghadge, D. G. Mogale, M. Bourlakis, L. M. Maiyar, and H. Moradlou, "Link between Industry 4.0 and green supply chain management: Evidence from the automotive industry," *Computers & Industrial Engineering*, vol. 169, p. 108303, Jul. 2022, doi: 10.1016/j.cie.2022.108303.

[27] V. Sharma, R. D. Raut, M. Hajiaghaei-Keshteli, B. E. Narkhede, R. Gokhale, and P. Priyadarshinee, “Mediating effect of industry 4.0 technologies on the supply chain management practices and supply chain performance,” *Journal of Environmental Management*, vol. 322, p. 115945, Nov. 2022, doi: 10.1016/j.jenvman.2022.115945.

[28] Ö. Başak and Y. E. Albayrak, “Petri net based decision system modeling in real-time scheduling and control of flexible automotive manufacturing systems,” *Computers & Industrial Engineering*, vol. 86, pp. 116–126, Aug. 2015, doi: 10.1016/j.cie.2014.09.024.

[29] J. Ding, H. Sun, X. Chen, and H. Fang, “Response time analysis of a manufacturing supply chain with performance evaluation process algebra,” *Computers & Industrial Engineering*, vol. 167, p. 108043, May 2022, doi: 10.1016/j.cie.2022.108043.

[30] H. Sonar, A. Gunasekaran, S. Agrawal, and M. Roy, “Role of lean, agile, resilient, green, and sustainable paradigm in supplier selection,” *Cleaner Logistics and Supply Chain*, vol. 4, p. 100059, Jul. 2022, doi: 10.1016/j.clscn.2022.100059.

[31] P. Gohoungodji, A. B. N’Dri, J.-M. Latulippe, and A. L. B. Matos, “What is stopping the automotive industry from going green? A systematic review of barriers to green innovation in the automotive industry,” *Journal of Cleaner Production*, vol. 277, p. 123524, Dec. 2020, doi: 10.1016/j.jclepro.2020.123524.

[32] X. Zhang, “A Petri net based simulation to study the impact of customer response to stock-out on supply chain performance,” *Cogent Engineering*, vol. 3, no. 1, p. 1220112, Dec. 2016, doi: 10.1080/23311916.2016.1220112.

[33] P. Paulraj *et al.*, “Environmentally Conscious Manufacturing and Life Cycle Analysis: A State-of-the-Art Survey,” *Journal of Nanomaterials*, vol. 2022, pp. 1–17, Jul. 2022, doi: 10.1155/2022/8438462.

[34] K. E. Moore, A. Güngör, and S. M. Gupta, “Petri net approach to disassembly process planning for products with complex AND/OR precedence relationships,” *European Journal of Operational Research*, vol. 135, no. 2, pp. 428–449, Dec. 2001, doi: 10.1016/S0377-2217(00)00321-0.

[35] S. K. Srivastava, “Green supply-chain management: A state-of-the-art literature review,” *Int J Management Reviews*, vol. 9, no. 1, pp. 53–80, Mar. 2007, doi: 10.1111/j.1468-2370.2007.00202.x.

[36] X. Zhang, Q. Lu, and T. Wu, “Petri-net based applications for supply chain management: an overview,” *International Journal of Production Research*, vol. 49, no. 13, pp. 3939–3961, Jul. 2011, doi: 10.1080/00207543.2010.492800.

[37] P. Ahi and C. Searcy, “A comparative literature analysis of definitions for green and sustainable supply chain management,” *Journal of Cleaner Production*, vol. 52, pp. 329–341, Aug. 2013, doi: 10.1016/j.jclepro.2013.02.018.

[38] C. R. Carter and D. S. Rogers, “A framework of sustainable supply chain management: moving toward new theory,” *Int Jnl Phys Dist & Log Manage*, vol. 38, no. 5, pp. 360–387, Jun. 2008, doi: 10.1108/09600030810882816.

[39] M. Kadziński, T. Tervonen, M. K. Tomczyk, and R. Dekker, “Evaluation of multi-objective optimization approaches for solving green supply chain design problems,” *Omega*, vol. 68, pp. 168–184, Apr. 2017, doi: 10.1016/j.omega.2016.07.003.

[40] M. El-Sayed, N. Afia, and A. El-Kharbotly, “A stochastic model for forward–reverse logistics network design under risk,” *Computers & Industrial Engineering*, vol. 58, no. 3, pp. 423–431, Apr. 2010, doi: 10.1016/j.cie.2008.09.040.

[41] L. Y. Y. Lu and C. H. Wu, “Environmental principles applicable to green supplier evaluation by using multi-objective decision analysis,” p. 15.

[42] M. Dotoli and M. P. Fanti, “Modeling and Performance Evaluation of a Supply Chain via Petri Nets,” p. 8.

[43] N. Viswanadham and N. R. Srinivasa Raghavan, “Performance analysis and design of supply chains: a Petri net approach,” *Journal of the Operational Research Society*, vol. 51, no. 10, pp. 1158–1169, Oct. 2000, doi: 10.1057/palgrave.jors.2600063.

[44] S. Allesina, A. Azzi, D. Battini, and A. Regattieri, “Performance measurement in supply chains: new network analysis and entropic indexes,” *International Journal of Production Research*, vol. 48, no. 8, pp. 2297–2321, Apr. 2010, doi: 10.1080/00207540802647327.

[45] A. Nagurney and L. S. Nagurney, “Sustainable supply chain network design: a multicriteria perspective,” *International Journal of Sustainable Engineering*, vol. 3, no. 3, pp. 189–197, Sep. 2010, doi: 10.1080/19397038.2010.491562.

[46] C.-M. Lorena and R.-U. Leonardo, “Sustainable procurement with Coloured Petri Nets. Application and extension of the proposed model,” *Expert Systems with Applications*, vol. 114, pp. 467–478, Dec. 2018, doi: 10.1016/j.eswa.2018.07.043.

[47] R. I. van Hoek, “From reversed logistics to green supply chains,” *Supply Chain Management: An International Journal*, vol. 4, no. 3, pp. 129–135, Jan. 1999, doi: 10.1108/13598549910279576.