electronics

MDPI

Article

Simulation in the GPenSIM Environment of the Movement of Vehicles in the City Based on Their License Plate Numbers

Tomasz Kossowski [1,*], Sławomir Samolej [2] and Reggie Davidrajuh [3]

[1] Department of Electrical and Computer Engineering Fundamentals, Rzeszow University of Technology, 35-959 Rzeszów, Poland
[2] Department of Computer and Control Engineering, Rzeszow University of Technology, 35-959 Rzeszów, Poland
[3] Department of Computer and Electrical Technology, University of Stavanger, 4036 Stavanger, Norway
* Correspondence: t.kossowski@prz.edu.pl; Tel.: +48-17-865-1298

Abstract: This paper uses colored Petri nets (PN) to develop a model of vehicle movement in a city. The modeler defines the number and location of crossroads, the connections between them, and how many vehicles are at the given points of the network. The vehicles are recognized by their license plate numbers, and it is possible to determine where they start their journey and where they are going. The algorithm proposed in this paper suggests the shortest vehicle route based on Dijkstra's algorithm. This study focuses on improving route planning by considering road usage, which is determined by the starting and ending locations of vehicles (as if traffic cameras were identifying license plates). This approach will lead to optimal control of traffic lights (or vehicle navigation) to minimize traffic jams and make good use of all roads. Additionally, this paper shares the results of preliminary simulations using both colored and uncolored Petri nets in the GPenSIM environment.

Keywords: discrete system simulation; colored Petri nets; traffic lights; GPenSIM

Citation: Kossowski, T.; Samolej, S.; Davidrajuh, R. Simulation in the GPenSIM Environment of the Movement of Vehicles in the City Based on Their License Plate Numbers. *Electronics* **2024**, *13*, 683. https://doi.org/10.3390/electronics13040683

Academic Editors: Valentina E. Balas, Dariusz Kania and Andrzej Pułka

Received: 20 December 2023
Revised: 22 January 2024
Accepted: 4 February 2024
Published: 7 February 2024

1. Introduction

Petri net is a tool for modeling discrete systems. Petri net was invented in the 1960s by Carl Adam Petri. Petri nets are closely related to automaton theory, but they have the ability to express concurrent events [1]. In its simplest version, a Petri net consists of places, transitions, and directed arcs. Such a network can only describe a system as a static combination of possible states [2,3]. To describe a particular state of the system, "tokens" are needed, which can be moved between places through transitions along the graph's arcs. In their simplest form, tokens in a Petri net are indistinguishable from one another [2]. More extended forms of Petri nets use concepts such as token coloring, transition activation time, and hierarchy.

This paper uses an extended Petri net—a colored Petri net—to develop a city model for the dynamic simulation of vehicle traffic. Colored Petri net (CPN) distinguishes tokens using colors, where color is an identifier with a specific name [4,5]. This identifier can be the same for multiple tokens (e.g., "blue") or individuals as a registration number (each token has a different "color" as an identifier, e.g., 001, 002, 003, and so on) [6,7].

In a Petri net, the most important elements are:

- Places: Places are drawn as circles, inside which the token presented by the black dots can be placed. There can be any non-negative number of tokens in one place.
- Transitions: Transitions are drawn with rectangles.
- Arcs: Arcs can have weights greater than or equal to 1. Weights equal to 1 (default value) are not shown in the Petri net. The weight determines exactly how many tokens pass along the arc [8].

There are some problems associated with modeling real-life discrete event systems using Petri nets:

- Petri net models are very large (number of data and structures) for real-life systems, even for simple systems. Such models are not small or compact [9].
- Long simulation time: Simulation of Petri nets takes a long time as large matrix manipulations are involved (as a large Petri net model results in a large incidence matrix). In addition, every enabled transition is checked to determine whether it can start firing by checking the additional firing conditions.
- Difficulty in model analysis: Due to the huge size, analyzing structural and behavioral properties becomes time-consuming.
- "Explosion of states": Possibility to obtain information (for certain Petri net classes) from the Petri net structure without exploring its state space. The Petri net tool automatically generates the state space, showing every possible state that can eventually be reached from the initial state. For real-life systems, this space is huge (and usually, it is of infinite size). Hence, analyzing such a huge or infinite state space is difficult and usually impossible [10–13].

Many tools are available for Petri net-based modeling (e.g., CPN and GPenSIM). Some extensions increase the computational power of modeling while maintaining its analytical capabilities (e.g., colored Petri nets) [10,13–15].

A manual description of crossroads throughout the city with the definition of dependencies and transit trigger conditions would be impossible. It is, therefore, necessary to develop crossroad modules as universal blocks. Modular Petri nets are proposed in some works [9,13,16,17] (e.g., for control of traffic lights [18]) and implemented in the General-purpose Petri net Simulator (GPenSIM) [13]. Modeling, simulation, and analysis of modular Petri nets using GPenSIM offer many possibilities, e.g., all input data can be collected in the form of tables, and the model structure can be built automatically from modules as needed. This type of approach saves time and avoids errors when describing each state [13,15,16,19].

Managing traffic signals through an algorithm that adapts to traffic volume has been known for many years. The most commonly used for this are induction loops in the roadway, counting the queue of vehicles [20]. Newer solutions include CCTV video surveillance with object recognition [21,22]. This system works very well in each part of the day but is sensitive; e.g., errors arise when objects move too close together and when weather conditions are unfavorable (such as rain and snow). Currently, the solutions are based on machine learning and neural networks, allowing us to achieve better results compared to classical algorithms [23]. The network learns the traffic characteristics of a given crossroad and optimizes traffic routing. Combining crossroads throughout the city into a single system allows for even better management of traffic signal control. Systems based on artificial intelligence are already being used and tested in China [24,25]. More and more, algorithms are not only able to identify objects (like cars and people), but they can also discern and interpret vehicle license plates. Such data analysis at each crossroad can identify the exact route taken by each vehicle within the range of the CCTV system. The extent to which this information can aid in traffic optimization is yet to be determined. Based on these data, can we better regulate traffic light cycles, thereby directing cars to roads with less traffic? Alternatively, will the information be transmitted to the vehicles so that the driver can take the route with a lower ETA (estimated time of arrival)? This solution (combined with a mesh network of vehicles and city infrastructure) could be particularly relevant for autonomous vehicles. There are a few possible applications for the proposed solution. The security of distributed systems, where the infrastructure is connected directly to users, is also a significant issue, as an open network is exposed to attacks, which poses a major challenge to ensuring its stability and resilience against cyber attacks [26]. At this stage, the primary focus is to introduce the concept and carry out its simulation, which is the main objective of this study.

The optimization of urban traffic with the help of proper algorithms that determine the cycles of traffic lights has been known for years [20,21]. Hence, this study proposes a novel approach, in addition to the existing ones, where alterations in the paths of moving vehicles

are facilitated by modifying their directions in the navigation system. This proposition is based on the concept of suggesting routes with a minimal estimated time of arrival (ETA), taking into account the prevailing traffic conditions. A significant difference from current navigation (through, e.g., online maps, which have such functionality) is the use of information from traffic cameras for this purpose. It is not navigation that is supposed to decide which route is the shortest at any given time, but the integrated management system of the city's traffic light network. Knowing information about the typical (most often duplicated) routes taken by individual vehicles each day (based on reading license plate numbers and recording the path of each vehicle—one of this paper's assumptions), it would be possible to propose access routes in such a way that centrally controlled traffic is optimally distributed. Machine learning, or AI algorithms, which are now being extensively used, could be used for this purpose (in further research). As previously stated, such a groundbreaking control system would be especially useful in self-driving vehicles. By recognizing the recurring start and end points of a journey (considering that most of us follow a similar route to and from work daily) and correlating them with consistent travel times, it becomes feasible to design optimal routes, which may not necessarily be the shortest. This would enable vehicles to move in a "green wave", minimizing their waiting time in traffic jams at intersections. Currently, such solutions are not used due to the enormous complexity of the proposed problem. It would be necessary to integrate the entire system of cameras in the city (they are not everywhere, which makes the task more difficult), along with a database of routes traveled (optimized by an adaptive algorithm), and end with the indication of the route to individual (willing) users. At this moment, such a system can be created only virtually as a simulation; however, its development, together with the demonstration of practical advantages, can be realized in practice in the future. This paper aims to focus on developing a model of the intersection network along with a description of the required properties. The goal is to conduct simulations on the fully functioning model before implementing optimization; conducting advanced optimization is proposed as a continuation of this research (further work).

The proposed model aims to completely change the plane and level of traffic control. The basic axiom changes because we no longer want to control traffic lights in such a way as to relieve traffic jams. We want to control traffic and vehicles so that these jams do not form or are as small as possible. This is a completely different approach from the current one, and ongoing simulation studies are leading to proof of this thesis.

The most important goals are:

- Development of crossroad and road modules (algorithms and their construction in a simulation environment).
- Proposing the concept of a modular colored Petri net to solve the given problem.
- Comparison of the possibilities of a colored and uncolored network based on the assumptions of universal modules.
- Allowing the user to define the appearance of the city and crossroads, as well as the number of vehicles, in a transparent and convenient way.
- Visualization of the above data in graphical form.
- Development of functions that automatically generate data and parameters necessary to perform simulations in the GPenSIM environment.
- Using this tool to create a model and simulate sample input data.

The rest of the paper is organized as follows: The basic concepts of Petri nets are introduced in Section 2. Section 3 describes a P/T (noncolored) Petri nets alternative example. Section 4 shows the results of the simulation using a colored Petri net. Finally, Section 5 presents conclusions and future research.

2. Concept of Modular Colored Token Petri Net (MCTPN)

Research on optimizing vehicle routing using colored Petri nets requires some assumptions. Of course, there are many works where authors have proposed various assumptions and solutions using Petri nets (colored and timed) [27–30]. However, none of them use the

capabilities of GPenSIM, and there are very few where modularity has been used. The goal, therefore, is to develop such a network so that it can be modeled and shown from a different, previously unknown perspective. The proposed solution is not only a description of the modular Petri net itself (as a mathematical model) but is also a tool that can be extended in further works. The end goal is to optimize the control of traffic signals or vehicles to make both the travel time as short as possible and the distribution of traffic balanced. Controlling the movement of cars can be performed based on correcting navigation indications as they pass through successive crossroads. Such corrections can be applied using data from the central traffic management system. The second solution is the automatic rerouting of an autonomous vehicle. Vehicles of this type will be on the roads more and more every year. There are already about 40 million autonomous cars (and around 1.5 billion of all) in the world, with a projected growth of 5 million per year [21,26,31]. In both cases, we can talk about automatic route control either by the vehicle's behavior or by directing it with the appropriate traffic light setting. This automation system, however, must be managed by a complex computer algorithm. The possibilities are many, and this article proposes the use of the GPenSIM tool and the Matlab environment.

For modeling large systems as modular systems, the modular Petri net is defined as a to-tuple (Equations (1)–(3)) [12–14]:

$$\mathrm{MCTPN} = (\mathrm{C}, \mathrm{S}) \tag{1}$$

where

$$\mathrm{C} = \sum_{i=1}^{m} \Phi_i \ (\mathrm{Crossroads} - \mathrm{Petri\ Modules}) \tag{2}$$

$$\mathrm{S} = \sum_{j=1}^{n} \Psi_j \ (\mathrm{Street} - \mathrm{connectors\ between\ Petri\ Modules}) \tag{3}$$

The formal definition of a single module (MCTPN) is shown in Formula (4), and the connector between modules is in Formula (5). A detailed description of the function of each component will be discussed when the colored Petri net algorithm is described in detail in Section 3 [16,17].

$$\Phi = (\mathrm{P_I}, \mathrm{P_O}, \mathrm{T_C}, \mathrm{T_G}, \mathrm{A_C}, \mathrm{M_C}) \tag{4}$$

where:

$\mathrm{P_I}$—input places of the crossroad;
$\mathrm{P_O}$—output places of the crossroad;
$\mathrm{T_C}$—traffic light transitions at the crossroad;
$\mathrm{T_G}$—colored token transitions of the crossroad;
$\mathrm{A_C}$—arcs in the crossroad;
$\mathrm{M_C}$—initial markings in the crossroad.

$$\Psi = (\mathrm{T_S}, \mathrm{A_S}, \mathrm{M_S}) \tag{5}$$

where:

$\mathrm{T_S}$—transitions between two crossroads;
$\mathrm{A_S}$—arcs in the street (towards crossroads);
$\mathrm{M_S}$—initial markings in the street.

The implementation of the presented algorithm is based on the following assumptions:

1. **Colored:** each vehicle (token) has a unique number (license plate).
2. **Modular:** the network simulates the layout of crossroads in the city.
3. **Model define:** The user can define what the city looks like. Data can be prepared in a spreadsheet. Users can also define the number of cars at each point.
4. **Constant speed:** It means that vehicles cannot overtake each other.
5. **Shortest path:** the cars follow the shortest path to their destination (defined randomly). The Dijkstra algorithm was used for this.

6. **Single-lane roads:** all of them. Tokens congregate in a place in front of the traffic light and drive in any direction independently. There are no dedicated lanes for going straight or turning in either direction.
7. **No reverse:** no vehicle can turn around at the crossroad. This situation is impossible due to the determination of the shortest paths using Dijkstra's algorithm.
8. **Definition of routes:** the routes of the cars are known based on the readings of traffic cameras (in reality). For simulation, this information will be asked from the database (see point 5).
9. **Reconfigurable:** the simulation conditions may change each time. This does not include the starting and ending points of the vehicles, which are fixed. This means that vehicles may start at slightly different moments in time and that signals operate differently during each simulation.

The proposed modules are universal. It is possible to implement multi-lane roads [32], but this is not necessary for the proposed solution. In the proposed model, the road is not differentiated by the number of lanes, so it can be treated as both single-lane and multi-lane. This is due to the averaging of traffic, not due to the volume in front of the traffic lights themselves but relative to the road as a general section between crossroads—taking into account its full capacity. Scheme of whole modules was shown in Figure 1.

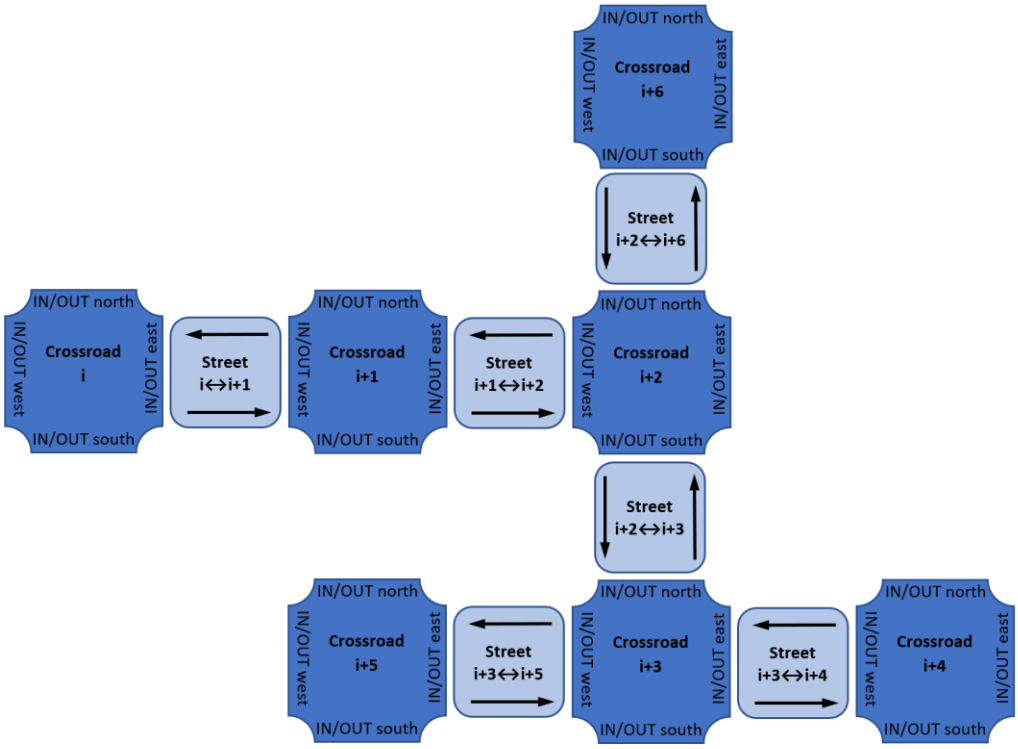

Figure 1. Scheme of modules representing segments of the city.

The following issues are key when using colors in GPenSIM [33,34]:

1. Only transitions can manipulate colors; output token colors can be added, removed, or changed in the preprocessor.
2. By default, the system collects all the colored tokens from the input locations when a transition is fired and then moves the colored tokens to the output locations.
3. An enabled transition can select specific input tokens based on color.
4. An enabled transition can select specific input tokens based on time (e.g., when the creation time of the tokens is known).
5. The token has the following structure: tokID, creation time, and color setting.

tokID: A single token identifier (integer value).

time_creation: The transition time (real value) when the token was created.

t_color: The color setting (the set of text strings).

Several functions are available in GPenSIM for color manipulation. One of those used in this article is *tokenEXColor*, which can be described as follows [9,13]:

[set_of_tokenID,nr_token_av] = tokenEXColor (place, nr_tokens_wanted, t_color), where the function requires three input arguments and returns two output values.

Input arguments *(place, nr_tokens_wanted, t_color):*

Place: From which place the tokens are to be selected.

nr_tokens_wanted: The number of required tokens of a certain color.

t_color: The set of colors.

Output values *[set_of_tokID,nr_token_av]:*

set_of_tokID: A set of tokIDs that meet the color specifications.

nr_token_av: Number of valid tokIDs available in set_of_tokID.

The crossroads module consists of four identical groups of elements (for each direction of the world). The full structure of the entire module will be presented in Section 3. However, it is important to explain the principle of operation of a single group in the block diagram shown in Figure 2. Its base is the transit system responsible for the passage of vehicles in a given direction. Its output is the place marked as an exit (along with information about the world direction and the number of the current crossroad). The inputs, on the other hand, are three. They correspond to the other directions of the world (except the one described for the exit, satisfying the assumption of no turning back). Tokens accumulate at the input locations and wait for a transit to be triggered (which corresponds to the situation at the actual crossroad). Each token has a different "color", and its routes are known. Based on these parameters, the transit only lets through tokens consistent with their destination direction. The remaining tokens wait for other transitions (corresponding to the other world directions, e.g., "north") to be triggered because the same inputs are connected by paths to the other transitions, as will be shown in detail in Section 3.

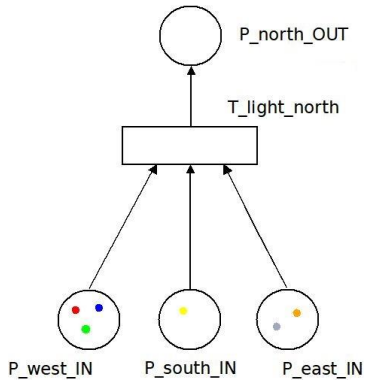

Figure 2. Scheme of one side of the traffic light (one form of four parts of a cross).

As mentioned, the most important element here is the transit using information about which tokens from which locations can be passed on. These parameters are generated automatically based on the following information:

1. Vehicle registration number (color);
2. starting point (random);
3. end point (random);
4. the shortest path, containing information on consecutive crossroad numbers.

Based on this information, a database is created where the following parameters are specified:

1. Crossroad number combined with the direction of exit (according to geographical direction);

2. registration numbers of vehicles to go in this direction to reach the next crossroad according to their shortest path;
3. the direction of entry (from where they are coming).

These data define the parameters necessary for the *COMMON_PRE* [13] function, making it possible to direct tokens using their colors. This is carried out by specifying the parameter *tokID1 = tokenAllColor (input buffer, number of tokens to pass, colors of tokens to pass)* [7,19]. These would not be needed in a classical Petri net, where there is no possibility of targeting tokens according to their colors (no vehicle differentiation). All parameters are calculated automatically from the input data set by the user. A block diagram of the function that generates the data matrix named **sp2** (**s**hortest **p**ath **2**) is shown in Figure 3.

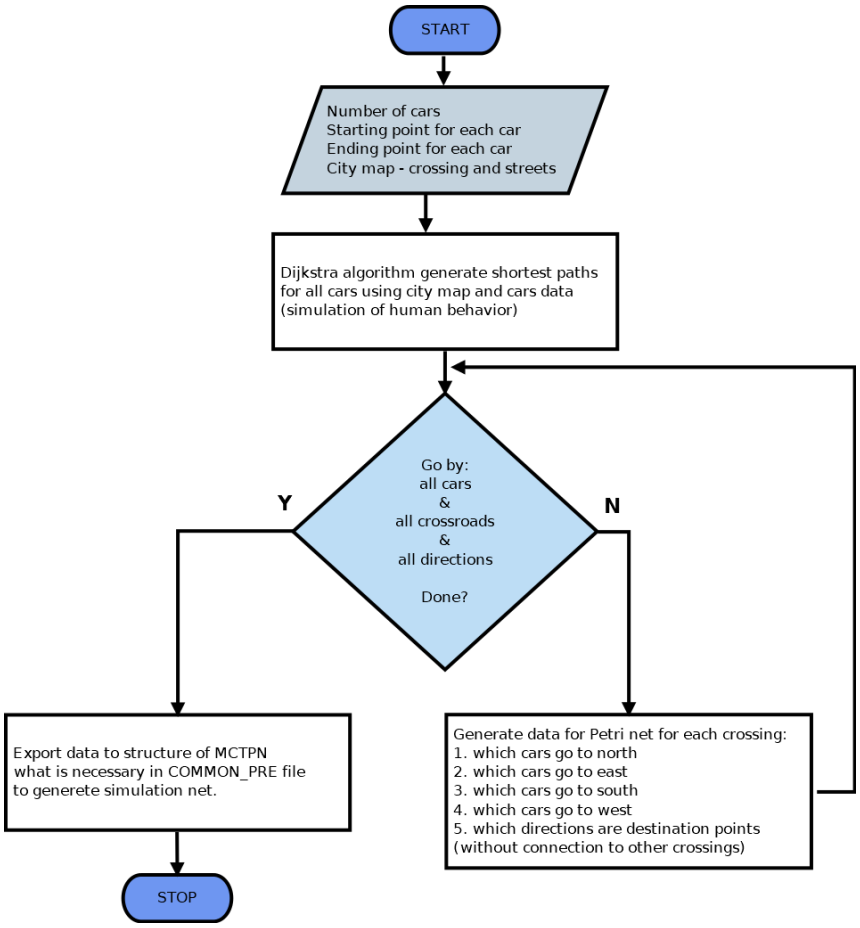

Figure 3. Block diagram of the algorithm that generates data for the COMMON_PRE.m file.

The very large loop Is due to the need to correlate data with the following:

1. All crossing;
2. for each crossroad i, the others are all crossroads j to which traffic is going ($i \cong j$);
3. all directions;
4. all shortest paths from the **sp** (**s**hortest **p**ath) matrix;
5. information on whether the vehicle is moving on or pulling over to the destination parking lot.

The data should be properly stored in the correct cells to minimize the number of operations in the *COMMON_PRE* function. The sp2 matrix generation function described above is executed once during the entire simulation, while the *COMMON_PRE* function is called for each transit for each fire. Therefore, it is important to perform most of the complex operations beforehand. Ready-made data will enable the entire simulation to run

much faster because the *COMMON_PRE* function, having ready-made data sets, does not have to search and calculate them every time.

Special transit blocks were used to generate the tokens. They are responsible only for the "departure" of vehicles from parking places directly adjacent to crossroads. In different simulation variants, the timing of their triggering (token release) can be changed independently of the operation of the crossroad transitions. This means that vehicles can set off quickly and jam up the city. Each of the transitions, labeled *T_gen*, has been implemented with a designation of its location—crossroad number and geographic orientation. A block diagram of the crossing segment containing vehicle-generating transitions is shown in Figure 4. The generated tokens can be retrieved by the transit agency responsible for directing traffic when they are triggered. These tokens are subsequently transferred to output locations to reach the next crossroads.

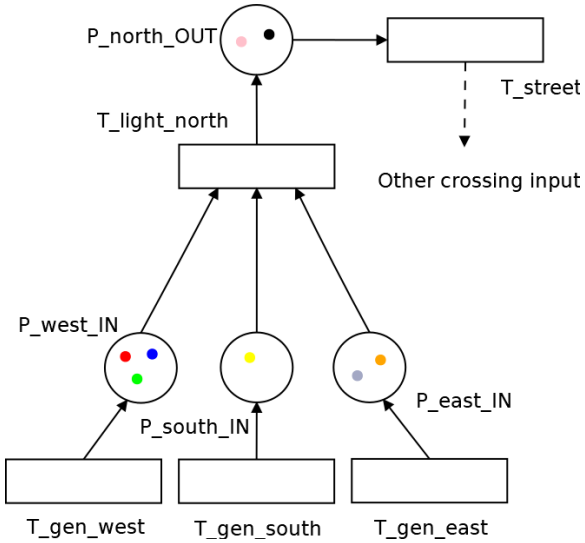

Figure 4. Scheme of tokens (i.e., cars) generation part—one section.

3. Colored Petri Net Modeling Algorithm

Using the described concept of a modular colored Petri net and the defined fragments of structures, it is possible to create entire functional blocks. Recall that these are the two basic types on which the entire simulated structure is built:

- The junction module consists of the four fragments described earlier.
- The road module that connects crossroads located directly next to each other is in the crossmap file input.

Both of these modules, together with all their components, are shown in Figure 5.

Individual sections of crossroads responsible for passing vehicles (tokens) in one of the world directions collect data from other directions (entrance places). For example, a transition directing vehicles to the north takes tokens from the input places in the directions east, west, and south. Therefore, the vehicle cannot turn around at the crossroad. This is due to the assumption that such an operation is not optimal from the point of view of the shortest path between the start and end points. With such a crossroads network structure, there is no way for Dijkstra's algorithm to route traffic through the same node twice (returning from the next and returning to the previous one to change direction). Each crossroad module has the same set of four entry and four exit locations, regardless of the connections between them. Entrances not connected to the road are treated as car parks adjacent to a given crossroad, from which vehicles can set off on the road (e.g., parking under a block of flats with access to the first crossroad). Exits without further connections are treated identically but as the end of the vehicle's journey (e.g., under the factory). In the case of connections with other crossroads, these places are points where vehicles move

along the road module. Decisions to target individual tokens were described in the previous section. However, in Figure 5, the transitions responsible for coloring the tokens (assigning them registration numbers and releasing them for traffic in the city) are not included due to the readability of the structure. The complete structure with all components is shown in Figure 6. The algorithm written in pseudocode can be presented as follows:

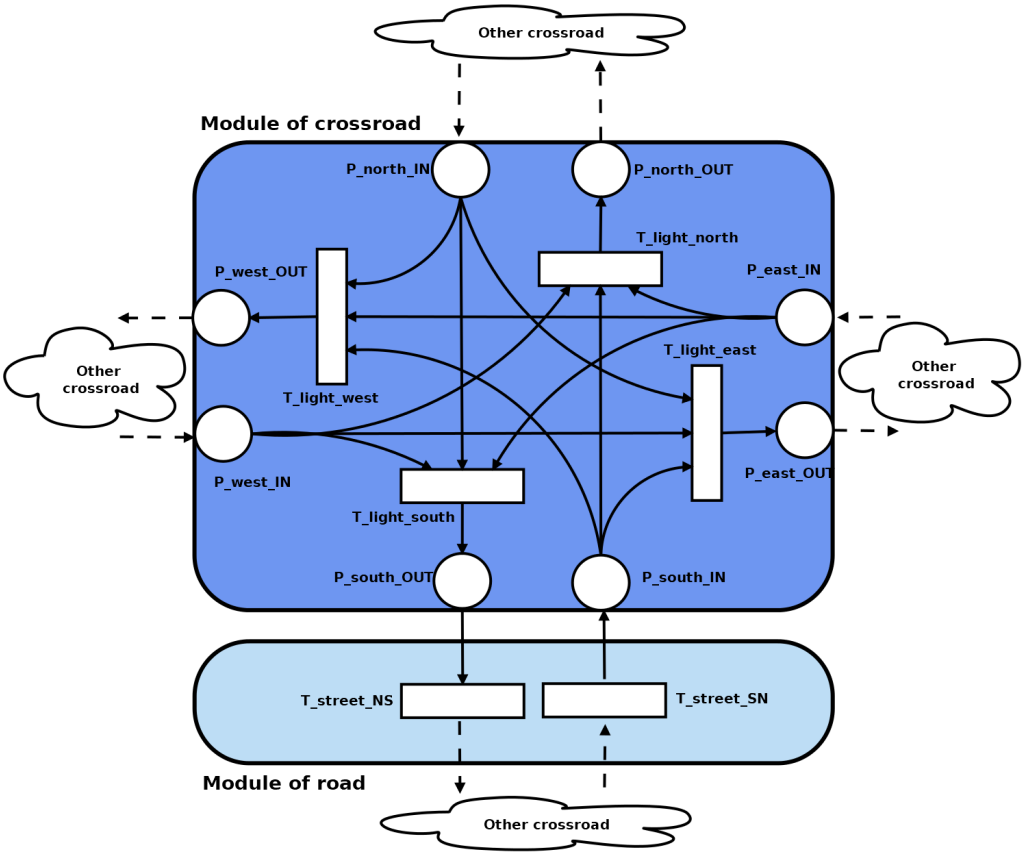

Figure 5. Modules of the colored Petri net block (without T_gen transitions).

The road module is generated automatically only when two crossroads are directly adjacent. The names of these modules are generated from the number of interconnected crossroads; in particular, transitions connecting traffic lanes in each direction are described in this way. On this basis, it is possible to clearly define a lane connecting two crossroads. For example, the lane connecting junctions 4 and 5 is *T_street_45*, but the return lane is *T_street_54*. The numbering is unique and completely distinguishable in the system, so you can adjust the parameters individually for each road (or globally for all types) at once. The global setting of parameters, such as firing time, is divided into each module separately. You can set this parameter for all crossroads, for all roads, and, if necessary, individually for each transition by modifying the matrix containing all the transitions in the model. Similarly, you can globally set the weights of all paths or independently of paths inside crossroads and roads. In the case of complicating the model, each path can be defined individually because all paths, along with their weight as a parameter, are available in the matrix of paths.

Main program pseudocode.

```
START
    User sets data: Car number, topography of city map, locations of parkings, and
    numbers of cars in each parking place.
    For each crossroad from MAP
        For each direction
            Generate transitions, places, and arcs in the CROSSROAD module
    For each pair of crossroads
        For each direction
            Generate transitions, places, and arcs in the STREET module,
            connected to CROSSROADS modules
    OR
        If it is not a pair
            Make connections to parking places
    For each car
        Generate random starting and ending points
        Generate the shortest path using the Dijkstra algorithm
        For each crossroad
            For each direction
                Generate data for transitions on Petri net about where moving
                tokens (cars) are
                Put tokens in places
                Set Arcs throughput
        Set all data for Transitions, Places, Arcs, firing time, simulation time,
        initial values, and others for Petri net in the GPenSIM library as a
        structure
            Generete colored Petri net model in GPenSIM
                Make simulation
                    Show results (defined by the user)
STOP
```

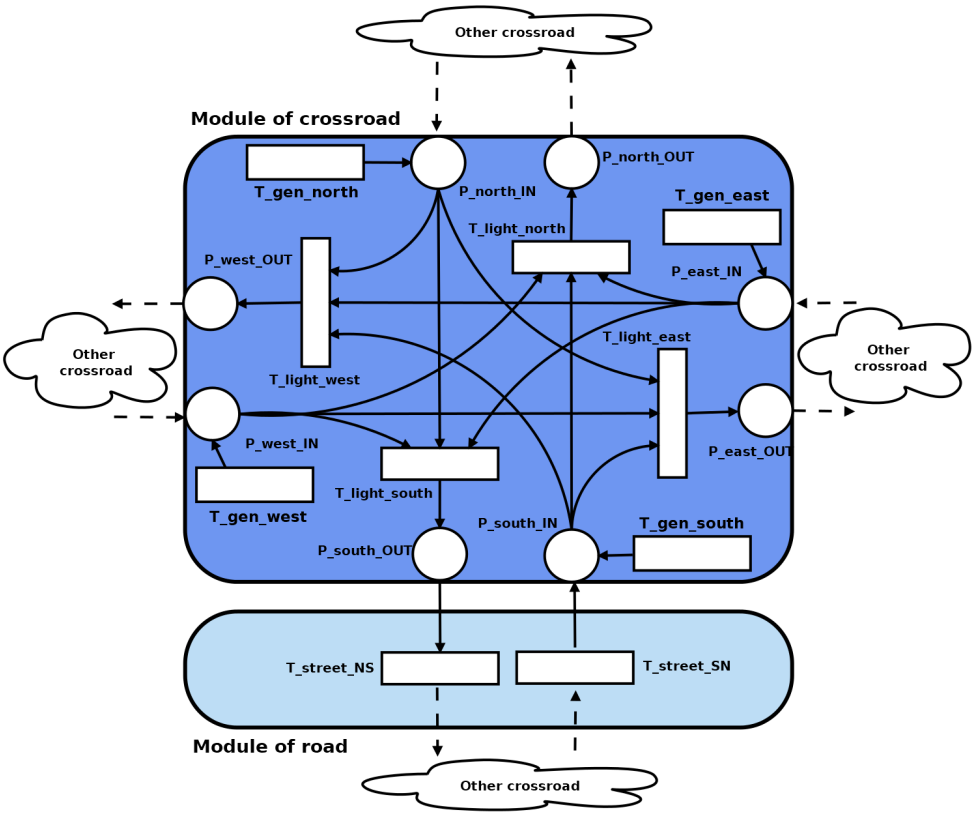

Figure 6. Complete modular colored Petri net block with whole elements.

4. Noncolored Petri Net Modeling Algorithm

The task of optimizing travel time for vehicles using Petri nets can be solved in two ways. The first assumes colored networks, where the "color" is the vehicle registration number. This consideration has already been described in the introduction. The second option assumes that there is no information about the starting and ending points of vehicles. Therefore, the analysis is carried out with a classic Petri net, where vehicles are treated as mere tokens. Due to the lack of information, this approach requires a simpler model and less decision-making. The algorithm is much simpler, resulting in much lower computational complexity and shorter computation time. An example of an algorithm using an uncolored Petri net is described in this chapter as an example of an alternative approach. The process of traffic management can be performed only by optimizing the traffic signal control system without interfering with the routes of moving vehicles. An example of the algorithm is shown in Figure 7.

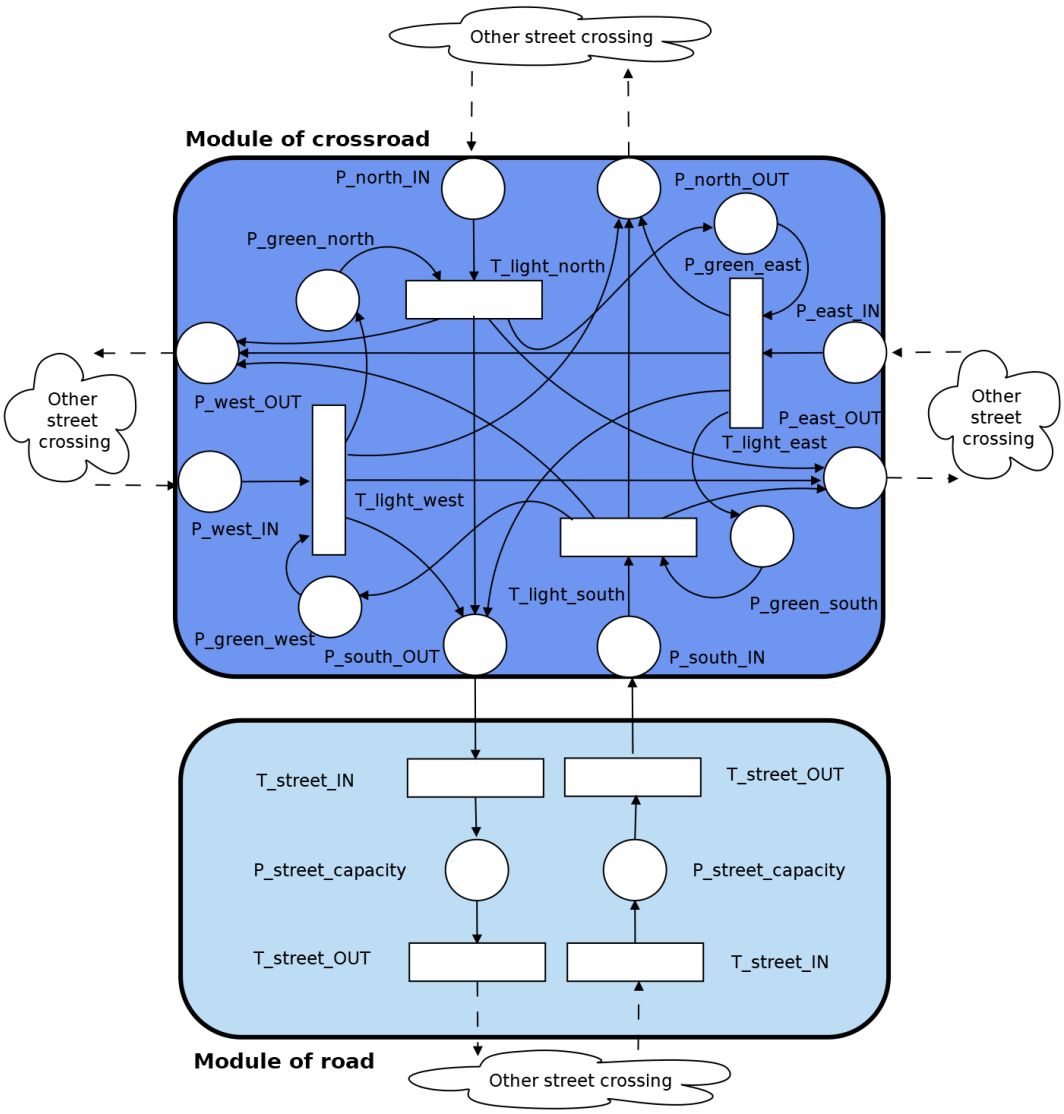

Figure 7. Noncolored Petri net block.

5. Results

Based on the presented algorithms and modules of crossroads and roads, software was developed as a function called in the GPenSIM (v. 9) environment. For the simulation to work correctly, it is necessary to prepare the main simulation file (MSF) that generates a Petri net model based on the input data. This file has the structure shown in Figure 8.

```
global global_info % user data
global_info.STOP_AT = 5;   % stop at time
run("crossgen.m"); %run generator of all data form input files

pns = pnstruct('cross_pdf'); % create petri net structure

    for i=1:length(ttime)
        if(mod(i,2)~=0)
            dyn.ft{i} = convertContainedStringsToChars(ttime(i));
        else
            dyn.ft{i} = str2double(ttime(i));
        end
    end

pni = initialdynamics(pns, dyn);
sim = gpensim(pni); % perform simulation runs

prnss(sim); % print the simulation results
figure(3);
plotp(sim, {'p_o15 west', 'p_o15 nord','p_o15 east','p_o15 south'});
```

Figure 8. Main simulation file.

The main simulation file references a number of other functions. Most of them are elements of the GPenSIM environment, but it is necessary to prepare a PDF file (Petri net Definition File) to build the model. In this file, all locations, places, and transitions are defined. This can be carried out manually or by creating appropriate functions that define all parameters from the input data, as was done. The PDF file is called from MSF as a function that returns the Petri net model. The data for the PDF file is generated in the crossgen.m file, which is the main component of the study. As described earlier, it generates all the data necessary for all the other functions that make up the model, which is ultimately simulated from the function called in MSF (*sim = gpensim(pni)*). The result of the generator is not only the simulation parameters but also a graphical map of crossroads in the city defined in the file by the user (Figure 9) and a graph where each crossroad is a node and the connections between them are one-way paths (Figure 10). In Figure 9, the x and y axes are the cardinal directions. The starting point of the city starts in the northwest and leads through subsequent points with intersections (placed on the grid). Figure 10 only shows the connections between intersections using the path model. You can therefore see possible connections, but without their geographical location.

The result for the sample input data is shown in Figure 9. Map of the city with crossings defined by the user in a spreadsheet. Simulation results for different times are shown in Figure 11 (20 units) and Figure 12 (50 units). These graphs illustrate the number of vehicles (tokens) at different locations. The graph in Figure 11 shows all geographic directions of one selected crossroad. It can be seen that there are no vehicles in the west direction; this is because there is no vehicle parking at this "location"—it is off-limits to traffic, so no vehicles arrive there. It can be seen that there is an increase in vehicles both in the parking lot located to the south and in the other directions, where this crossroad is connected to the others (numbered 13 and 16).

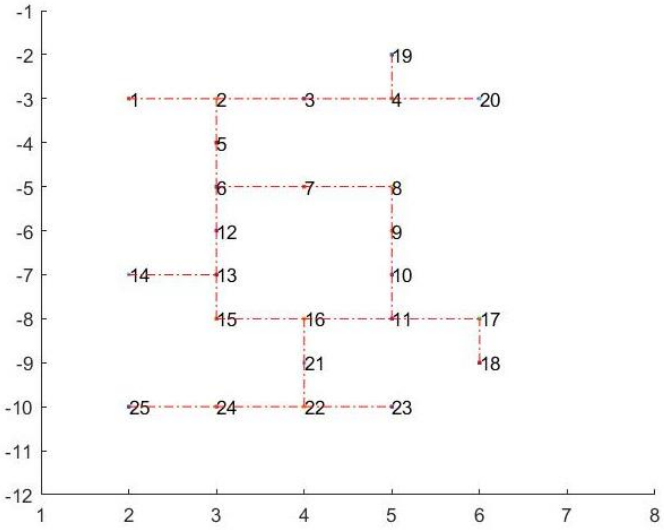

Figure 9. Map of the city with crossings defined by the user in a spreadsheet.

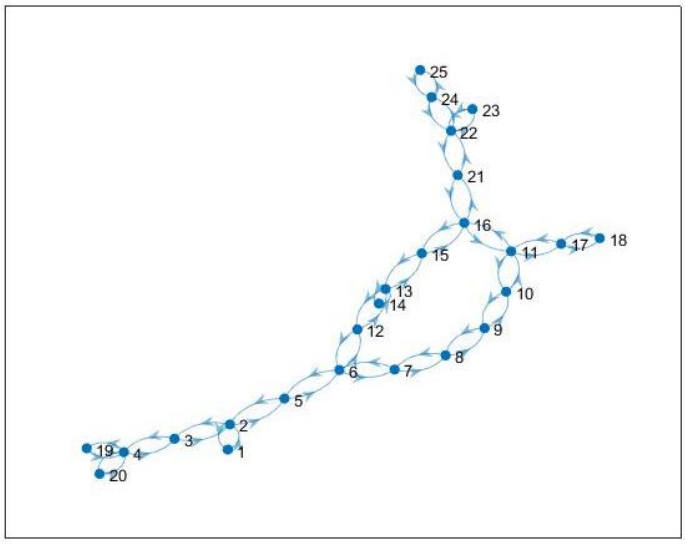

Figure 10. Diagram of the city with crossings used to define the shortest paths using Dijkstra's algorithm.

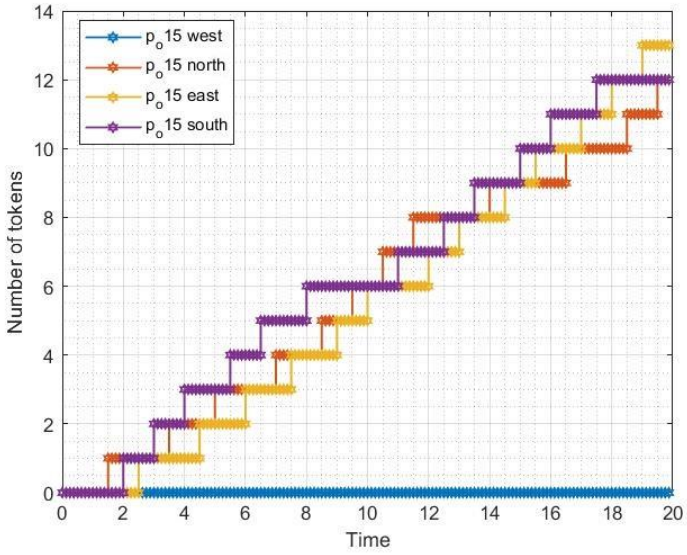

Figure 11. Results of the simulation.

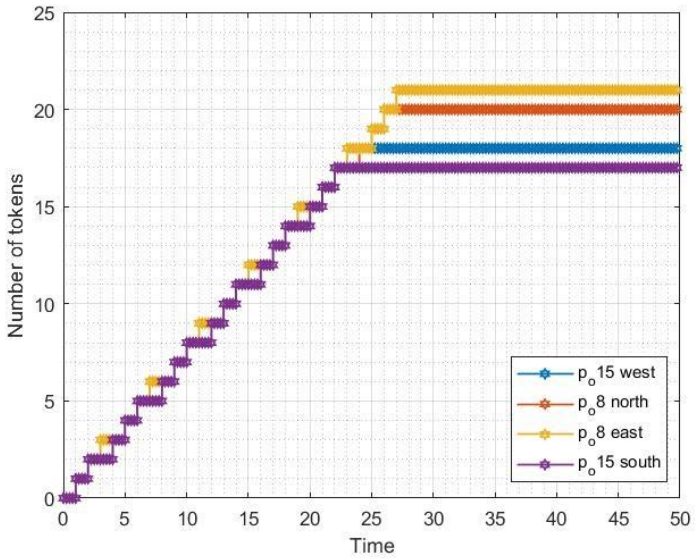

Figure 12. Results of the simulation (cars in selected ends' parking places).

Figure 12, in turn, shows the attainment of maximum vehicle values for the parking lots in question. This simulation had different input data, as cars could reach any point. Therefore, both parking lots for crossroad number 15 (west and south) accumulate tokens (unlike the earlier example). This graph only shows cars reaching their destinations at the two selected crossroads (numbers 8 and 15). Reaching the limit resulted from limiting the number of vehicles to 2000 for the entire city, which consists of 25 crossroads, as shown in Figure 9. The final quantities of tokens vary because their endpoints were selected randomly by the function that generates points for each car (discussed earlier). Both simulations, therefore, involve the same Petri net layout but differ in the initial conditions and simulation time. Figures 11–13 show the number of vehicles reaching randomly selected locations on the map (these may be parking lots or intersection exits). The x-axis presents the time flowing in the system, and the y-axis presents the number of vehicles at each moment in time. In the initial phase, traffic begins to increase and stabilizes as the road capacity becomes saturated, which can be seen for longer time intervals (e.g., Figure 12). For parking lots, it will reach the set value for the number of vehicles that were supposed to arrive there. For intersections, it will start to decrease over time (of course, if the number of cars leaving the parking lots starts to decrease).

Figure 13 shows the result of a modular simulation of a colored Petri net simulated in GPenSIM, showing the difference in the number of cars traveling on the road and arriving at the final destination (crossroad number 23 north). It can be clearly seen that the number of cars from the neighborhood arriving at the parking lot is clearly higher than those present in traffic. These are, of course, simulations for a short duration due to the computational complexity of the current solution. However, they show that the software produces results that can be further analyzed, especially considering further model development.

Simulations with longer times are very time-consuming, and optimization of some functions is still required to reduce the number of iterations while the simulation is running. Future changes will introduce optimizations that reduce computation time and add new functions for further research and simulation.

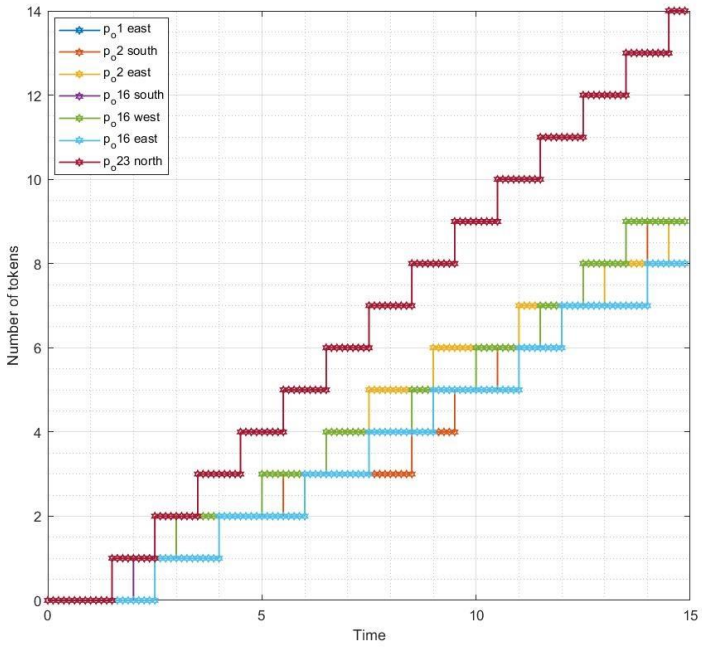

Figure 13. Results of simulation for cars on selected routes between crossroads and one (for 23 North) for an end parking place.

6. Discussion

There are many proposals for modeling the vehicle traffic system based on Petri net models. Among them, we can distinguish those based on classic, hybrid, modular, or colored networks [27–30,35–47]. There are detailed models of traffic lights taking into account their control and optimization [20,32,38], but none of them is based on recognizing each vehicle by its registration number. The reason may be the lack of infrastructure to analyze such a case in real-world conditions. However, other results can be indicated (improve traffic control) compared to those presented in this article, as presented in Table 1.

Table 1. Petri net methods for optimization traffic control.

Reference Number	Type of PN	Contributions
[27]	CTPN	A model of a real intersection to optimize the timed–colored Petri net for traffic control.
[38]	HPN	Solve the problem of coordinating several traffic lights with the aim of improving the performance of some classes of special vehicles, i.e., public and emergency vehicles.
[39,40]	DTPN	Minimize the number of vehicles in the network in PN (mathematical).
[41]	HPN	Applied model predictive control to predict the number of vehicles (optimizing algorithm).
[42]	CPN	A model with optimal routes for vehicles can be planned independently.
[43]	PA	Traffic control is dependent on current density due to the traffic system's mechanisms being dependent on a fixed time.
[44,45]	p-timed PN	Implement traffic-responsive control by extending green time on main roads.
[46]	TSPN	Models reduce the system complexity in terms of combinatorial explosion, and they could be adapted easily for any real intersection.
[47]	TPN	Optimization methods to manage the timing of traffic lights at intersections and speed limitations.

7. Conclusions

This article describes the concept of a modular colored Petri net. The GPenSIM tool operating in the MATLAB environment was selected for the simulation. Simulation of a small city can be performed by manually entering data into simulation files, but it is very time-consuming and carries a high risk of errors. It was decided to develop functions that automatically prepare all data and parameters to build the proper simulation model. Input files in the form of spreadsheets are used for this, where the user enters data (the city topology, crossroads, and the number of vehicles at each point). These data are processed when the main simulation file is run. The developed and discussed algorithms and functions create a Petri net module based on two basic modules: crossroads and roads. Coloring the tokens made it possible to distinguish the vehicles due to their registration numbers, which makes them completely distinguishable in the model. Thus, they can determine the start and end points of the route. There is a clear difference between a colored and an uncolored Petri net, especially in the same example described in the previous sections. The inability to distinguish between tokens allows traffic optimization, but without the ability to individually direct vehicles to the optimal path (due to temporary road congestion).

It can therefore be concluded, based on the above examples, that the uncolored Petri net allows traffic optimization by changing the behavior of the signaling system. The colored network, on the other hand, allows the same, transfers data directly to vehicles, and influences them to optimize their movement. This can practically be performed through rerouting in the navigation system or, in the case of autonomous vehicles, through dynamic re-routing in real-time. Of course, in practice, it is possible to combine both of these properties at the same time. All this will be the subject of further research on the proposed traffic optimization solution using modular colored Petri nets.

Further work will focus primarily on the already-mentioned optimization and the development of a mechanism to minimize the travel time for vehicles. It will therefore be crucial here to minimize the ETA parameter discussed earlier. Timed–colored Petri nets could be used for this purpose. They have the advantage of being able to control the moments of time when tokens reach specific locations and direct them through transitions relative to their lifetimes (or arrivals). The lifetimes of the tokens will also be a clear indicator of how long the journey took, making optimization and comparison of results more reliable [35,36].

The greatest achievements include the following:

- Development of crossroad and road modules (algorithms and their construction in a simulation environment).
- Proposing the concept of a modular colored Petri net to solve the given problem.
- Comparison of the possibilities of a colored and uncolored network based on the assumptions of universal modules.
- Allowing the user to define the appearance of the city and crossroads, as well as the number of vehicles, in a transparent and convenient way.
- Visualization of the above data in graphical form.
- Development of functions that automatically generate data and parameters necessary to perform simulations in the GPenSIM environment.
- Using this tool to create a model and simulate sample input data.
- Proposing further work related to the process of optimizing the control of both traffic lights and autonomous vehicles (or navigation in conventional vehicles).

In the proposed variant, we no longer want to control traffic lights to relieve jams. We want to prevent them. This is a completely different approach to the proposed solution based on the model of a modular colored Petri net. The presented and discussed software can be used for many different applications because it allows you to simulate traffic at crossroads in the city using distinguishable tokens. It is, therefore, a fully working solution that will be further developed.

Author Contributions: Conceptualization, T.K.; methodology, T.K. and S.S.; software, R.D. and T.K.; validation, T.K.; formal analysis, T.K. and S.S.; investigation, T.K.; resources, R.D. and T.K.; data curation, T.K.; writing—original draft preparation, T.K.; writing—review and editing, S.S. and R.D.; visualization, T.K.; supervision, T.K. and R.D.; project administration, T.K.; funding acquisition, T.K. All authors have read and agreed to the published version of the manuscript.

Funding: Research is funded by the University of Stavanger and by the Ministry of Science and Higher Education of the Republic of Poland to maintain the research potential of the disciplines of automation, electronics, electrical engineering, and space technologies.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Droste, M.; Shortt, R.M. From Petri Nets to Automata with Concurrency. In *Applied Categorical Structures*; Kluwer Academic Publishers: Alphen aan den Rijn, The Netherlands, 2002; Volume 10, pp. 173–191.
2. Jensen, K.; Kristensen, L.M.; Wells, L. Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. *Int. J. Softw. Tools Technol. Transf.* **2007**, *9*, 213–254. [CrossRef]
3. Peterson, J.L. *Petri Net Theory and the Modeling of Systems*; Prentice Hall PTR: Upper Saddle River, NJ, USA, 1981.
4. Rzonca, D.; Rząsa, W.; Samolej, S. Consequences of the form of restrictions in Coloured Petri Net models for behaviour of arrival stream generator used in performance evaluation. In Proceedings of the 25th International Conference, CN 2018, Gliwice, Poland, 19–22 June 2018; Computer Networks; Proceedings 25. Springer International Publishing: Cham, Switzerland, 2018; pp. 300–310.
5. Wang, C.; Feng, X.; Li, X.; Zhou, X.; Chen, P. Colored petri net model with automatic parallelization on real-time multicore architectures. *J. Syst. Archit.* **2014**, *60*, 293–304. [CrossRef]
6. Riemann, R.C. *Modelling of Concurrent Systems: Structural and Semantical Methods in the High Level Petri Net Calculus*; Herbert Utz Verlag: Munchen, Germany, 1999.
7. Davidrajuh, R. *Modeling Discrete-Event Systems with Gpensim: An Introduction*; Springer International Publishing: Cham, Switzerland, 2018.
8. Reisig, W. *Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies*; Springer: Berlin/Heidelberg, Germany, 2013.
9. Davidrajuh, R. Modular Petri net models of communicating agents. In Proceedings of the International Joint Conference SOCO'17-CISIS'17-ICEUTE'17, León, Spain, 6–8 September 2017; pp. 328–337.
10. Valmari, A. The state explosion problem. In *Advanced Course on Petri Nets*; Springer: Berlin/Heidelberg, Germany, 1996; pp. 429–528.
11. Clarke, E.; Grumberg, O.; Jha, S.; Lu, Y.; Veith, H. Progress on the state explosion problem in model checking. In *Informatics*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 176–194.
12. Baier, C.; Katoen, J.P. *Principles of Model Checking*; MIT Press: Cambridge, MA, USA, 2008.
13. Davidrajuh, R. A New Modular Petri Net for Modeling Large Discrete-Event Systems: A Proposal Based on the Literature Study. *Computers* **2019**, *8*, 83. [CrossRef]
14. Jensen, K. Coloured petri nets. In Proceedings of the IEE Colloquium on Discrete Event Systems: A New Challenge for Intelligent Control Systems, London, UK, 4 June 1993.
15. David, R.; Alla, H. *Discrete, Continuous, and Hybrid Petri Nets*; Springer: Berlin/Heidelberg, Germany, 2005.
16. Davidrajuh, R.; Joseph, J.F. Towards Modeling Road Tunnels: A Petri Nets based Approach. *Int. J. Simul. Syst. Sci. Technol.* **2022**, *23*. [CrossRef]
17. Riouali, Y.; Benhlima, L.; Bah, S. Petri net extension for traffic road modelling. In Proceedings of the IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA), Agadir, Morocco, 29 November–2 December 2016; pp. 1–6.
18. Huang, Y.S.; Weng, Y.S.; Zhou, M. Modular design of urban traffic-light control systems based on synchronized timed Petri nets. *IEEE Trans. Intell. Transp. Syst.* **2013**, *15*, 530–539. [CrossRef]
19. Davidrajuh, R. *Petri Nets for Modeling of Large Discrete Systems*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 95–105.
20. Wiseman, Y. Traffic Light with Inductive Detector Loops and Diverse Time Periods. *Contemp. Res. Trend IT Converg. Technol.* **2016**, *4*, 166–170.
21. Mulay, S.; Dhekne, C.; Bapat, R.; Budukh, T.; Gadgil, S. Intelligent city traffic management and public transportation system. *arXiv* **2013**, arXiv:1310.5793.
22. Eamthanakul, B.; Ketcham, M.; Chumuang, N. The traffic congestion investigating system by image processing from CCTV camera. In Proceedings of the 2017 International Conference on Digital Arts, Media and Technology (ICDAMT), Chiang Mai, Thailand, 1–4 March 2017; pp. 240–245.
23. Peppa, M.V.; Bell, D.; Komar, T.; Xiao, W. Urban traffic flow analysis based on deep learning car detection from CCTV image series. In *SPRS TC IV Mid-Term Symposium "3D Spatial Information Science–The Engine of Change"*; Newcastle University: Newcastle upon Tyne, UK, 2018.

24. Joo, H.; Lim, Y. Intelligent traffic signal phase distribution system using deep Q-network. *Appl. Sci.* **2022**, *12*, 425. [CrossRef]
25. Shi, Y.; Wang, Z.; LaClair, T.J.; Wang, C.R.; Shao, Y.; Yuan, J. A Novel Deep Reinforcement Learning Approach to Traffic Signal Control with Connected Vehicles. *Appl. Sci.* **2023**, *13*, 2750. [CrossRef]
26. Feng, Y.; Huang, S.E.; Wong, W.; Chen, Q.A.; Mao, Z.M.; Liu, H.X. On the Cybersecurity of Traffic Signal Control System with Connected Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 16267–16279. [CrossRef]
27. Dotoli, M.; Fanti, M.P. An urban traffic network model via coloured timed Petri nets. *Control Eng. Pract.* **2006**, *14*, 1213–1229. [CrossRef]
28. Tang, J.; Piera, M.A.; Guasch, T. Coloured Petri net-based traffic collision avoidance system encounter model for the analysis of potential induced collisions. *Transp. Res. Part C Emerg. Technol.* **2016**, *67*, 357–377. [CrossRef]
29. List, G.F.; Cetin, M. Modeling traffic signal control using Petri nets. *IEEE Trans. Intell. Transp. Syst.* **2004**, *5*, 177–187. [CrossRef]
30. Wang, J. *Timed Petri Nets: Theory and Application*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012; Volume 9.
31. Statista. Available online: https://www.statista.com/statistics/1230664/projected-number-autonomous-cars-worldwide (accessed on 20 March 2023).
32. Luo, J.; Huang, Y.S.; Weng, Y.S. Design of variable traffic light control systems for preventing two-way grid network traffic jams using timed Petri nets. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 3117–3127. [CrossRef]
33. Kaid, H.; Al-Ahmari, A.; Li, Z.; Davidrajuh, R. Intelligent Colored Token Petri Nets for Modeling, Control, and Validation of Dynamic Changes in Reconfigurable Manufacturing Systems. *Processes* **2020**, *8*, 358. [CrossRef]
34. Kaid, H.; Al-Ahmari, A.; Li, Z.; Davidrajuh, R. Single Controller-Based Colored Petri Nets for Deadlock Control in Automated Manufacturing Systems. *Processes* **2020**, *8*, 21. [CrossRef]
35. Samolej, S.; Szmuc, T. Time Constraints Modeling And Verification Using Timed Colored Petri Nets. In *Real-Time Programming 2004*; Elsevier: Amsterdam, The Netherlands, 2005; pp. 127–132.
36. Bevilacqua, M.; Ciarapica, F.E.; Giovanni, M. Timed coloured petri nets for modelling and managing processes and projects. *Procedia CIRP* **2018**, *67*, 58–62. [CrossRef]
37. Ng, K.M.; Reaz, M.B.; Ali, M.A. A review on the applications of Petri nets in modeling, analysis, and control of urban traffic. *IEEE Trans. Intell. Transp. Syst.* **2013**, *14*, 858–870. [CrossRef]
38. Di Febbraro, A.; Giglio, D.; Sacco, N. Urban traffic control structure based on hybrid Petri nets. *IEEE Trans. Intell. Transp. Syst.* **2004**, *5*, 224–237. [CrossRef]
39. Di Febbraro, A.; Giglio, D. On adopting a Petri net-based switching modeling system to represent and control urban areas. In Proceedings of the 8th International IEEE Conference on Intelligent Transportation Systems, Vienna, Austria, 13–16 September 2005; pp. 185–190.
40. Di Febbraro, A.; Giglio, D. Traffic-responsive signaling control through a modular/switching model represented via DTPN. In Proceedings of the IEEE Intelligent Transportation Systems Conference, Toronto, ON, Canada, 17–20 September 2006; pp. 1430–1435.
41. Basile, F.; Chiacchio, P.; Teta, D. A hybrid model for real-time simulation of urban traffic. *Control Eng. Pr.* **2012**, *20*, 123–137. [CrossRef]
42. Liang, X.; Dang, Y.; Hou, Y. Modeling and Analysis of Urban Traffic System Based on Colored Petri Nets. In Proceedings of the 2021 IEEE International Conference on Networking, Sensing and Control (ICNSC), Xiamen, China, 3–5 December 2021; Volume 1, pp. 1–6.
43. Anas, A.M.; Terzioğlu, H.; Durdu, A. Intelligent Traffic Signaling Control Using Petri Nets. *Artif. Intell. Stud.* **2020**, *3*, 1–13.
44. Soares, M.S.; Vrancken, J. A modular Petri net to modeling and scenario analysis of a network of road traffic signals. *Control Eng. Pract.* **2012**, *20*, 1183–1194. [CrossRef]
45. Soares, M.S.; Vrancken, J. Responsive traffic signals designed with Petri nets. In Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics, Singapore, 12–15 October 2008; pp. 1942–1947.
46. Elidrissi, H.L.; Moh, A.N.S.; Tajer, A. Modular Design and Adaptive Control of Urban Signalized Intersections Systems Using Synchronized Timed Petri Nets. *Comput. Inform.* **2022**, *41*, 590–608. [CrossRef]
47. Mohammadi, M.; Dideban, A.; Moshiri, B. A novel approach to modular control of highway and arterial networks using petri nets modeling. *Int. J. Eng.* **2023**, *36*, 1578–1588. [CrossRef]