A Model of Software System for Parking Using Search Algorithms

Dražen Drašković*, Sanja Vukićević **
* School of Electrical Engineering, University of Belgrade, Serbia
** School of Organizational Sciences, University of Belgrade, Serbia
drazen.draskovic@etf.bg.ac.rs, vukicevicsanja@yahoo.com

Abstract - In this paper a model of software system for parking using search algorithms has been described. The basic idea is to provide a system user with a fast and simplified way to find the nearest empty place while entering the parking lot of large surface. During the searching procedure, modified branch and bound method is used.

On entering the parking lot, a system user is connected to a server by means of the Bluetooth. Then, the server sends the parking lot schema in a predefined XML format to a smart phone, as well as the identifier of the entrance where the user is currently located. Then, a parking schema is being created on the phone and a route to the closest parking space is being displayed. This software system has been developed to be used as an educational system for teaching the academic course 'Expert Systems', but it could be improved and used in real-life applications as well.

Keywords - Branch and Bound Search Algorithm, Educational System, Expert Systems, Parking System, Wireless Sensor Network (WSN)

I. Introduction

With the development of contemporary technologies, people very often do not show understanding of everyday activities, which may waste lot of time. However, with the help of modern technologies such activities may be accelerated to a great extent. Generally speaking, large parking lots with lots of parking space are used more worldwide. Usually, users see the number of empty parking places at the very entrance, but they are not able to notice the location of the nearest parking place. The situation, in which a user moving through the parking lot may find himself at the spot where he should turn left or right not knowing where he would find the parking place faster and more easily, represents a real problem. The borderline case may occur in which a user cruises endlessly within the parking lot while some parking places are being emptied, and then filled by other users who have been faster. According to the study by IBM in 20 cities worldwide drivers usually spend daily 20 minutes on average in order to find an empty parking place [1]. This time, spent by users trying to find a place, increases the petrol consumption, and the harmful gases release pollutes the environment.

This project is supported by the Ministry of Education and Science of the Republic of Serbia.

The idea of this paper is to create the system that would enable the users to find the first empty parking place more easily. At the entrance a user would obtain the schema of the whole parking lot via his mobile phone. Sending the parking lot schema may be fulfilled by means of some data transfer wireless technologies. Thus, on the basis of the obtained parking lot schema, a mobile phone would draw a path to the nearest parking place whereby a user reach the first empty place quickly and easily by following the drawn path. In addition to this, all empty places within the parking lot would be marked, which enables a user to select another empty place at his own will. This system would lead to minimizing the number of vehicles circulating around the parking lot in search of an empty place at parking with high level of congestion. The vehicle movement time at the parking lot would thus be significantly minimized, and the entering parking time would not be maximized, mainly because a user must wait for the entrance ramp to be lifted during which time a parking lot schema will be received via a mobile phone.

In section II, the technical requirements for the creation of this software system have been given, while Section III provides the problem description with the model parking lot description together with the search algorithm. Section IV provides the conclusion of this paper.

II. Technical Requirements

Certain parking lot technical requirements are needed for the proper functioning of the designed system. Figure 1 displays the technical requirements diagram for a parking lot and for a user, as well as their interconnection. The displayed devices on the technical requirements diagram are the following: (1) the central server, (2) the distributed terminal, (3) a Bluetooth terminal adaptor, (4) a user's mobile phone, and (5) parking place status sensors, respectively.

The central server and a user's mobile phone are connected by means of the distributed terminal and the Bluetooth terminal adaptor that represent intermediaries in communication. On the other hand, parking place status sensors have one-way connection with the server.

The central server represents a processing unit which keeps a record on all the parking places and their current availability for a user (a place is either empty or occupied). At a given moment, on the central server there is up-to-date status of parking spaces. In case that a parking lot has a great number of parking places, the parking place availability check-up may be a complex time consuming operation. At the same time, the central server is also a server of a distributed system, which is accessed by the terminals using a predefined protocol. Terminals send requirements for the parking schema to the central server. If a great number of terminals exist, then the central server must provide the parallel requirements processing in order to prevent the slowing down of the terminals in cases of great traffic jam. The system performances heavily depend on the central server performances.

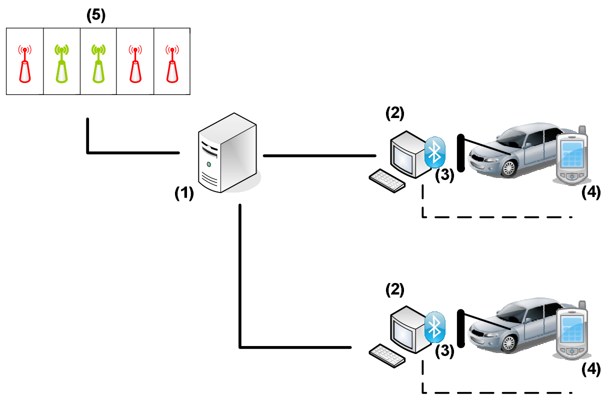

Figure 1. Parking Lot System Model

The distributed terminals represent the devices located at every entrance of a parking lot. A terminal serves as an intermediary in communication between a user and the central server. The need for terminal existence has arisen due to a reason that there are several entrances within the larger parking lots. In case of the central server non-existence, it would be necessary for the terminals to communicate between each other concerning the parking place availability, which renders the communication protocol of the whole system more complex and demanding, while simultaneously increasing the data synchronization time and leads to the terminal response delay. The central server and terminal connection must be carried out via fast connections. In the borderline case, the information flow up to the central server will be equal to the speed of terminal requests multiplied by the number of terminals. For this reason, a terminal should possess appropriate resources so that these pieces of information are being processed timely, and the response would be then sent back to the terminal.

A parking place status sensor is a device that enables the automatic detection of parking place availability. The sensor detects the presence of an object having vehicle dimensions whereby the oscillation between occupied and empty state is evaded too often. It could be a problem, if sensor detects a person who is passing past the sensor. This problem was solved by using two sensors on each parking place [2]. Status sensors are connected to the sensor network which communicates with the central server. For system implementation in real-life, we suggest using sensors explained in [3].

The communication between the sensor network and the central server must be stable and fast, because the parking place state update frequency will be proportional to the vehicle circulation at the parking lot [4]. If the vehicles circulation at the parking lot is extremely frequent, it might be the case that a great number of sensors would then send the parking place status information too frequently, thus making the central server overloaded and unable to process other requests coming from the terminal. The parking place status sensor request which is sent to the central server has the higher priority during the processing, because it is essential that all the terminals should be provided with the updated parking lot situation picture.

The Bluetooth adaptor is a device that has to exist at each terminal. The Bluetooth adaptor represents an intermediary at the physical level of communication between users, that is, between a mobile phone and a terminal, which passes the requests to the central server. The predefined protocol for the exchange of messages and other objects is being used. At each terminal an adaptor must be adjusted to be visible so that a user might connect by means of his mobile phone to the terminal he moves towards.

A user must possess a mobile phone with the application able to display a parking lot schema obtained by the Bluetooth communication with the terminal. A mobile phone must also support the Bluetooth protocol so that the mobile phone should be visible to the terminal with the Bluetooth adaptor. The application should be enabled to establish a device visibility so that an adapter can find it and approve the Bluetooth communication with other devices. The mobile phone must have empty memory space in order to permit the application to save the parking lot terminal information and display it to the user. Accordingly, the application should be allowed to enter data into the memory card and allowed to read data from that card too.

The new Android phones support Bluetooth Core version 2.0 with Enhanced Data Rate (EDR), for fast data transfer. The practical data transfer rate of EDR is 2.1 Mbit/s [5]. Considering the schema which is transferred via Bluetooth is in XML format, it does not require faster connection.

Users, who do not have smart phones, may also use this system but with modification of the proposed solution. The whole parking lot should possess the system of the Bluetooth transmitter and the Bluetooth receiver, and a mobile phone would be constantly connected with only one receiver/transmitter. In the ideal case it would always be the nearest one, because the signal of such a receiver/transmitter is the strongest one. The empty parking place notice would be sent to users in the format of a text message.

III. The Problem Description

Parking lot schema defining ought to be carried out before the system installation. By parking lot schema defining the system is being provided with the information concerning the parking spaces set up, the parking lot shape, the entrance/exit system, and the possibility of passing from one part of the parking lot to another one, and the like. XML files have been used for the parking lot schema description, and the system has been realized in the Java programming language, using the Android libraries.

A. The Parking Lot Model

The basic elements which represent a parking lot are the following:

- The parking block,
- The parking place,
- The parking entrance,
- The transition between the two parking blocks.

Every parking element has a unique ID for a group of elements to which it belongs. Parking places, entrances to the parking block, as well as the transitions between two parking blocks might be located only at the edge of a parking block. If the need arises for parking places in the middle of the rectangular parking lot, this would be achieved by merging several parking blocks. Parking blocks constitute the tree structure. The block which contains the entrance at which the user is located represents the tree root. The simplest way to write down this model is in the XML files format. The XML file possesses the main tag entitled <ParkingLotSchema> within which the other tags, described in the remaining part of the paper, are contained. The parking block represents a block within which parking places are located, and which may be accessed by using the parking entrance and from which one might move to the neighboring parking block by means of the transition between the two parking blocks. The parking block may be of any shape. At this moment, there exists a support only for the rectangular parking block. The rectangular parking block is described through its height, width, parent block, merge place in relation to the parent block

```
<ParkingBlock ID='1'>
     <Type>Rectangular</Type>
     <Width>15</Width>
     <Height>14</Height>
     <OffsetX>0</OffsetX>
     <OffsetY>15</OffsetY>
     <ParentBlock>3</ParentBlock>
     <MergingPoint>10</MergingPoint>
     <MergingSide>right</MergingSide>
     <ParkingSpots>
       <ParkingSpot>...</ParkingSpot>
     </ParkingSpots>
     <Gateways>
       <Gateway>...</Gateway>
     </Gateways>
     <Entries>
       <Entry>...</Entry>
     </Entries>
</ParkingBlock>
```

Figure 2. XML code for a parking block

and the merge side in relation to the parent block. If the parent block does not contain the data on the place of merging with the parent block and the side of merging in relation to the parent block, the data become irrelevant. The parking block contains the list of parking places which are located within it, the list of parking block entrances, as well as the list of transitions to the parent parking block. An example of one parking block, described in the XML file is provided in Figure 2.

The type of a given parking block is rectangular. The displayed block in the parking blocks tree has a parent block with ID=3, located at the parent right side, starting from the tenth meter of the parent right side (viewed from above downwards). Only the name of a new block should be added in case of introducing new shapes of parking blocks. The move absolute values change along the appropriate axes, even though they are given in the block structure. Equally, these values might be calculated in the application itself.

The parking place represents space aimed for one vehicle. The parking place is described by its height, width, position in the given block, the side at which it is located within the parking block and the status, which might be either empty or occupied. An example of a parking place, described in the XML file, is given in Figure 3. A parking place located at the lower part of the given parking block is displayed, as well as the initial point of parking place branching which is on the coordinate 0, and the block is extended 3 meters to the right and 5 meters upwards. The current status of this parking place is „occupied“.

```
<ParkingSpot>
     <BlockSide>Bottom</BlockSide>
     <StartPoint>0</StartPoint>
     <Width>3</Width>
     <Height>5</Height>
     <Status>Occupied</Status>
</ParkingSpot>
```

Figure 3. XML code for a parking places

The parking block entrance is described in terms of its width, position and side at which it is located within the appropriate parking block. An example of one such parking entrance, described in the XML file, is given in Figure 4. The displayed example represents the parking block entrance located on the left side of the given parking block, being 3 meter wide and the branching point from which it begins is located on the left side of the block, 5 meters from its upper point. Since the entrance is located on the left side of the block, or more precisely on one of the vertical sides, the block entrance is extended according to the height, so the final extension point is at the eighth meter from the upper side of the block on its left side.

```
<Entry>
     <EntrySide>left</EntrySide>
     <Width>3</Width>
     <StartPoint>5</StartPoint>
</Entry>
```

Figure 4. XML code for entry point

The transition between two parking blocks represents the place to which one might move from one to parking block to another. The transition between the two parking blocks is described in terms of the width, position in an appropriate parking block, as well as the side on which it is located in an appropriate parking block. Currently, every transition is a two-way one. Nevertheless, it might be introduced in the subsequent versions of the system that the transition type is one-way or two-way transition. An example of one transition between the two parking blocks, described in the XML file, is given in Figure 5.

```
<Gateway>
      <GatewaySide>Top</GatewaySide>
      <Width>5</Width>
      <StartPoint>10</StartPoint>
</Gateway>
```

Figure 5. XML code for gateway

This system supports all sorts of parking that may be described by a sequence of mutually perpendicular and parallel rectangles which are mutually in the contact. The most basic shape is a rectangular parking lot, which contains parking places along its edge. Such a parking lot is described by only one block at which parking places are located. If a parking lot shape is a rectangular and has parking places not only along the edge, but also within the block, a need would arise for the existence of several parking blocks, which are lined up one upon the other.

Figure 6 illustrates two blocks. Block 1 represents the root block in the parking tree. Block 2 immediately follows the first one. The merge side for block 2 is the lower one, because it is viewed in relation to the parents block. The place of merging equals 0 and blocks have the same width (merge side). Block 2 contains two transitions to the parent block. Both transitions are located at its bottom side, one begins at the beginning and the other one at the end of the bottom side. The first two rows of parking places belong to the upper block, and the other two to the lower block. Each block contains two parking

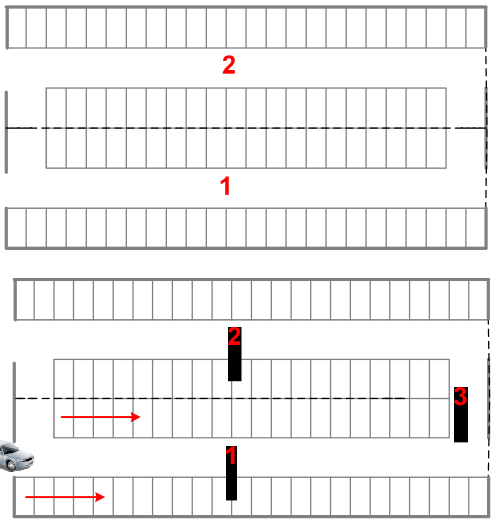

Figure 6. The different block schema for the same parking lot

entrances, from its left and right side. Provided that a vehicle enters through the left entrance, and the first empty place is in block 2, the schema by which it would move is a schema through the left transition to the second block to the parking place.

The implementation of the system itself is divided into classes for parking structure realization and the specific Android classes. The classes comprising the parking structure are the classes describing the block, the rectangular block, parking place, parking entrance, the transition between the parking blocks and the search for the nearest parking place on the given side of the parking block. Android classes realize the very appearance of the application, as well as the communication with the parking server.

B. The Implemented Algorithm

The XML scheme will serve the purpose of dynamic tree formation depending on the vehicle position. Figure 7 displays an example of the Faculty of Electrical Engineering parking lot schema. Graph in Figure 7 displays every schema block by node, while each transition between the blocks is represented by relationship. If two or more rectangular blocks meet, connections towards all the blocks will be defined.

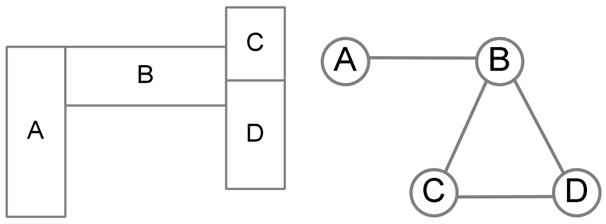

Figure 7. Blok schema and graph for parking of the Faculty of Electrical Engineering

The algorithm starts by locating the block in which the vehicle is situated (block A). Having detected the belonging to a particular block, a search in that block is being initiated as well as a search in all the neighboring blocks (block B). A search is performed in such a way that the shortest distance between the located vehicle coordinates and the parking place coordinates are searched for. The algorithm forms the search tree and memorizes the nodes which are visited during the search so that they do not move the same direction several times.

The search may be simultaneously performed in the observed and in the neighboring blocks. If an empty place has not been found in the neighboring place, it is being checked whether any block has already found the nearest parking place. If this is the case, the search is being finalized and stopped, and the nearest place is selected from among the found ones. If this is not the case, the search continues in its neighboring blocks, which have not been searched through. We initialize four threads, one for each block side, for the rectangular shape that is being searched through. On finishing all the search threads, each search thread reports on one parking place if there is one existent. Having received the report on four parking places, one is being selected that is the nearest one. The search for neighboring blocks is performed likewise. Whether an algorithm would find a better place in the neighboring blocks also depends on the transition distance between the observed block and its neighbor. If the nearest transition to the neighboring block is placed farther from the place we have found in the observed block, the search in the neighboring block is not necessary. As it is presented in Figure 6, blocks could be the same size but it is not usual in real life, as it can be seen in Figure 7. Each block contains information how many parking spots it consists. Therefore, time for block searching is different.

When a user enters the block 1 parking entrance displayed in Figure 6, the algorithm initiates the threads for both sides of the block. It is being inspected whether the parking places are empty or occupied in each thread for all the parking places. Then, the empty parking places, if there are such, sorts ascending in relation to the entrance distance. The thread execution result is the first empty parking place nearest to the entrance. When both threads finish the search, the empty parking places are compared and the one nearest to the entrance is selected, but the obtained parking place does not represent the ultimate result. Namely, it might be the case that in block 2 there is a parking place which is nearer from the one found in the block 1. Therefore, parallel with the block 1 threads, all the neighboring blocks of the block one are searched through, i.e. blocks 2 and 3. They are searched through in the same way as block 1. When the search in all three blocks is finished, the found parking places are compared and then the parking place nearest to the entrance is being proclaimed. Optimization may be performed in the algorithm, so that the search is not carried out in the block which is more distanced from the last parking place in the block within which a car is located. In this example, this would mean that the empty parking places search would not be performed in block 3, unless there is no empty place in blocks 1 and 2. In specific case, if driver isn't able to occupy empty parking spot, because another driver has occupied an empty parking spot at this time, the driver will require an empty parking spot again, through a Bluetooth adaptor.

IV. Future work

Possible improvement of the selected model are support for other types of parking blocks, one-way transition between parking blocks, more levels parking support, and the nearest parking exit search. Other shapes of parking blocks may be added to the system, in such a way as to define the block description and the new types of sides on condition that the edges are neither horizontal nor vertical. The block is being currently described only by its height and width, which is not sufficient enough when we describe, for example a triangularly-shaped parking lot. One-way transitions between the blocks are possible with addition of the additional data which would define whether the transition from one block to another is unidirectional, and if this is the case, which direction movement is permitted. Also, during the parking place search, the algorithm might include the direction transition check up. Modifications for parking with several levels is simple, because the transition from one block of the lower level to the higher level block would be regarded as ascending or descending. It would be also useful to add the information to the blocks concerning the level to which they belong, so that the system user can follow the movement on the drawn schema. The calculation of movements along the coordinates would be more complicated for parking lots with several levels than in the case of one level parking lot. Similarly, as there is a search for the nearest parking place from the entrance, it is possible to implement the nearest exit of the parking block. The search algorithm remains the same as in the case of empty parking place search, with the exception of finding the exit, which must be defined in the XML schema.

Defining the parking structure is possible also in the form of the weighted graph, in which case, instead of searching according to the branch and bound algorithm, algorithm A* would be implemented, taking into consideration also the schema weight, which would represent heuristics in searching [6]. This graph is useful in case when one transition is more passable than the other, so the former should be used, even though it is somewhat placed farther.

V. Conclusion

Our research is based on the standard parking analysis, represented by the structure of mutually perpendicular or parallel rectangles with two-way transition points between the blocks, and then, on building a software system model and defining the search algorithm.

The described system enables its users to save time in the course of finding the nearest available parking place. The existing parking system is not complicated to be upgraded. On the contrary, it is necessary to add the wireless infrastructure, in such a way as to place and adjust the terminal with the Bluetooth adaptor connected to the central server at every entrance ramp. Also, it is necessary to define the parking lot schema with the division into blocks. This might be complex depending on the parking size and complexity, but should be done only once. Finally, the most demanding process of parking updating represents installing parking sensors at every parking place. The complexity of parking installation is directly dependent on the parking lot size and the number of parking places.

Due to its modified search algorithm, this system is also used as an educational system in teaching process within the subject 'Expert Systems' for undergraduate academic studies.

References

[1] "IBM Global Parking Survey: Drivers Share Worldwide Parking Woes", IBM, New York, 2011. http://www-03.ibm.com/press/us/en/pressrelease/35515.wss, accessed on January 29th, 2012

[2] J. Chinrungrueng, U. Sunantachaikul, S. Triamlumlerd, "Smart Parking: an Application of optical Wireless Sensor Network", International Symposium on Applications and the Internet Workshops, Hiroshima, 2007.

[3] S. V. Srikanth, P. J. Pramod, K. P. Dileep, S. Tapas, Mahesh U. Patil, N. Sarat Chandra Babu, "Design and Implementation of prototype Smart PARKing (SPARK) Sysem using Wireless Sensor Network", International Conference on Advanced Information Networking and Applications Workshops, Singapore, 2009.

[4] S. Vujcic, G. Rakocevic, N. Kojic, D. Milicev, D. Vitas, "A Classification and Comparison of Data Mining Algorithms for Wireless Sensor Networks", The IEEE International Conference on Industrial Technology, Athens, 2012.

[5] G. Kewney, "High speed Bluetooth comes a step closer: enhanced data rate approved", November 16th, 2004, http://www.newswireless.net, accessed on March 22nd, 2012

[6] B. Nikolic, "Expert Systems", WUS Austria Educational Publishing and University of Belgrade, Classroom Textbook, June 2011.