**0.3 THE PROBLEM AND IMPORANCE OF SYSTEM**

There are many wastage instances of natural resources due to the cruising vehicles. Again the carbon footprint on natural biomes is also a considerable factor. Hence, an optimal strategy to find a parking spot can relieve traffic congestion, reduce air pollution and enhance driving comfortability.

Basically in the parking optimization is based on two costs.

**0.3.1.1 Individual Cost**

The penalty on an individual due to the parking problem. Includes Gas cost, wastage of time, exhaustion, unnecessary delay due to traffic congestion etc.

**0.3.1.2 Social Cost**

The penalty on our natural biomes due to the parking problem. Includes air pollution, carbon footprint, noise pollution etc.

**0.3.2 IMPORTANCE**

Smart Parking involves the use of low cost sensors, real-time data and applications that allow users to monitor available and unavailable parking spots. The goal is to automate and decrease time spent manually searching for the optimal parking floor, spot and even lot. Some top benefits of a parking solution are –

**0.3.2.1 Optimized Parking**

Users find the best spot available, saving time, resources and effort. The parking lot fills up efficiently and space can be utilized properly by commercial and corporate entities.

**0.3.2.2 Reduced Traffic**

Traffic flow increases as fewer cars are required to drive around searching for available parking spots.

**0.3.2.3 Reduced Pollution**

Searching for parking burns around 8.37 million gallons of gasoline per day. An optimal parking solution will significantly decrease driving time, thus lowering the amount of daily vehicle emissions and ultimately reducing global environmental footprints.

**0.3.2.4 Increased Safety**

Parking lot employees and security guards contains real-time lot data that can help prevent parking violations and suspicious activity. Also decreased spot-searching traffic on the streets can reduce accidents caused by the distractions of searching for parking.

**0.3.2.5 Real-Time Data and Trend Insight**

Over time, a smart parking solution can produce data that uncovers correlations and trends of users and lots. These trends can prove to be invaluable to lot owners as to how to make adjustment and improvements to drivers.

**0.3.3 WHAT HAS BEEN DONE**

Most advanced parking facilities available in INDIA provide only compact mechanical parking solutions that are based on APS (Automated Parking System). This system provides a vertically stacked parking system that lowers the space and volume consumption of a parking lot. But this type of system does not provide any assistance to the driver if the parking spot is filled. Again the driver goes back to cruise mode and hit-and-try method.

This system also does not maintain a balanced parking network. One APS maybe used to its full extent and others may remain under-used. So basically this type of system does not tackle either the individual cost problem or the social cost problem.

**0.4 SCOPE OF PROPOSED WORK**

**0.4.1.1 REQUIREMENTS IN GLOBAL SCENARIO**

* There are estimated 600 million cars in the world, a figure that’s rapidly growing, especially in china and other emerging market, shows us the vast population of cars and the necessity of parking systems.
* 30% of traffic congestion is caused due to vehicles in search for parking.
* A waste of 8.37 million gallons of gasoline and over 129000tons of CO2 emission are taking place due to improper parking systems and congestion.

**0.4.1.2 REQUIREMENTS IN INDIAN SCENARIO**

* 20 minutes are wasted by drivers in India in search for parking.
* 43% of total driving time is spent looking for a parking spot.
* A recent survey shows that during rush hours in most big cities, the traffic generated by cars searching for parking spaces takes up to 40% of the total traffic.

**0.4.2 TECHNICAL CHALLENGES**

Fishing for a parking space is a routine activity and the congestion is created by drivers for parking space resulting in loss one million barrels of world’s oil every day. Smart Parking solutions are designed to provide drivers an ultimate solution.

However, in our project of smart parking there are several challenges we will be facing:

**0.4.2.1 Increase in population of vehicles**

Rapid increase in population of vehicles surely creates a panic situation for parking in cities.So allotting a free parking space to the driver by providing them an optimised nearest location to park at a low cost will be a challenging option.

**0.4.2.2 Adopting Smart Parking Technology**

To change a culture which has been existing for centuries is a humungous task. So attracting people towards smart parking all of a sudden will be a challenging job.

**0.4.2.3 Enabling the tools and techniques**

To operate a method so vast and diverse the service providers use technological devices like RFID, databases to store updated information regularly, sensors, dynamic messaging systems, computer clients and servers, hardware drivers and application interface.

Enabling all these devices from thousands of vendors to communicate and tying them together into one platform is the greatest challenge.

**0.4.2.4 Controlling of congestion**

Depending upon the situation and time, congestion can arise across any parking area. So, designing an appropriate algorithm to avoid congestion as well as providing an option of nearest parking area will be a tough challenge.

**0.4.2.5 Countering the issues of over parking and low parking**

Maintaining a proper balance in parking across all parking areas also comes as a burning challenge.

**0.4.2.6 Fake Parking Requests**

As the user is allowed to book only limited parking spaces at a time from one id, it will decrease the chances for a single user to redundantly occupy many spaces.

**0.5 METHODOLOGY**

The project is designed as a reservation based system which has an algorithm to carry out a balanced parking network. Network in the sense that at a time more than one parking lots will be managed by this system. Here a vehicle re-directing algorithm is used at the time of reservation to perfectly distribute the vehicles registering for a parking space.

As for the design, it is assumed that the construction of the parking lot allows only one vehicle to pass at a time and the considered parking lot is a confined space, not a curb side parking spot.

The System architecture will have the need of the hardware components mentioned below:

**0.5.1 Parking Sensors**

These sensors will be deployed to sense an incoming or an outgoing vehicle. For our project we will be using Infra-red sensors. These sensors will be connected with either the microchip or an external terminal for their power supply.

**0.5.2 Central Processing Unit**

Arduino unit will be used as the main processing unit in this project which is a processor on chip. This processing unit will act as the intermediate between the sensors and the user application. All the sensors will be connected to the microprocessor.

**0.5.3 User Application**

This application will be meant and designed for the users. Using this application, users will be able to interact with the system to perform their required operation like checking for availability of vacant parking spaces or to reserve a spot for their vehicle etc.

**0.5.4 The Cloud**

The cloud is required to host the application to make it available to the users. By using the cloud services, the end users will be able to interact with the main system. It will act as a database to store the records related to the parking system and to make them available to the users. It will keep track of every user that are connected to the system.

**FIGURES OF MODULES USED**



CREATE MODULE



DISPOSE MODULE



DECIDE MODULE



PROCESS MODULE



DELAY MODULE



ROUTE MODULE



STATION MODULE

**0.6 MODULES USED IN SIMULATION**

**0.6.1 CREATE MODULE**

This module is intended as the starting point for entities in a simulation model. Entities are created using a schedule or based on a time between arrivals. Entities then leave the module to begin processing through the system. The entity type is specified in this module.

**0.6.1.1 TYPICAL USES**

* The start of a part’s production in a manufacturing line
* A document’s arrival (e.g., order, check, application) into a business process
* A customer’s arrival at a service process (e.g., retail store, restaurant, information desk)

|  |  |  |
| --- | --- | --- |
| Prompt | Valid Entry | Default |
| Name— Unique module identifier. This name is displayed on the module shape. | Symbol Name [All modules] | <module name and instance number> |
| Entity Type— Name of the entity type to be generated. | Symbol Name [Entity Names] | Entity 1 |
| Type— Type of arrival stream to be generated. Types include: Random (uses an Exponential distribution, user specifies mean), Schedule (uses an Exponential distribution, mean determined from the specified Schedule module), Constant (user specifies constant value, e.g., 100), or Expression (pull down list of various distributions). | Random (Expo), Schedule, Constant, Expression | Random |
| Value— Determines the mean of the exponential distribution (if Random is used) or the constant value (if Constant is used) for the time between arrivals. Applies only when Type is Random or Constant. | Real | 1 |
| Schedule Name— Identifies the name of the schedule to be used. The schedule defines the arrival pattern for entities arriving to the system. Applies only when Type is Schedule. | Symbol Name [Schedules] | Schedule 1 |
| Expression— Any distribution or value specifying the time between arrivals. Applies only when Type is Expression. | Expression (Distributions) | 1 |
| Units— Time units used for inter-arrival and first creation times. Does not apply when Type is Schedule. | Seconds, Minutes, Hours, Days | Hours |
| Entities per Arrival— Number of entities that will enter the system at a given time with each arrival. | Expression | 1 |
| Max Arrivals— Maximum number of arrivals that this module will generate. When this value is reached, the creation of new arrivals by this module ceases. | Expression | Infinite |
| First Creation— Starting time for the first entity to arrive into the system. Does not apply when Type is Schedule. | Expression | 0.0 |

**0.6.2 PROCESS MODULE**

This module is intended as the main processing method in the simulation. Options for seizing and releasing resource constraints are available. Additionally, there is the option to use a "submodel" and specify hierarchical user-defined logic. The process time is allocated to the entity and may be considered to be value added, non-value added, transfer, wait or other. The associated cost will be added to the appropriate category.

**0.6.2.1 TYPICAL USES**

* Machining a part
* Reviewing a document for completeness
* Fulfilling orders
* Serving a customer

|  |  |  |
| --- | --- | --- |
| Prompt | Valid Entry | Default |
| Name—Unique module identifier. This name is displayed on the module shape. | Symbol Name [All Modules] | <module name and instance number> |
| Type—Method of specifying logic within the module. Standard processing signifies that all logic will be stored within the Process module and defined by a particular Action. Submodel indicates that the logic will be hierarchically defined in a "submodel" that can include any number of logic modules. | Standard, Submodel | Standard |
| Action—Type of processing that will occur within the module. Delay simply indicates that a process delay will be incurred with no resource constraints. Seize Delay indicates that a resource(s) will be allocated in this module and a delay will occur, but that resource release will occur at a later time. Seize Delay Release indicates that a resource(s) will be allocated followed by a process delay and then the allocated resource(s) will be released. Delay Release indicates that a resource(s) has previously been allocated and that the entity will simply delay and release the specified resource(s). Applies only when Type is Standard. | Delay, Seize Delay, Seize Delay Release, Delay Release | Delay |
| Priority—Priority value of the entity waiting at this module for the specified resource(s). Used when one or more entities from other modules are waiting for the same resource(s). Does not apply when Action is Delay or Delay Release, or when Type is Submodel. | High (1), Medium (2), Low (3), Other Expression | Medium (2) |
| Resources—Lists the resources or resource sets used for entity processing. Does not apply when Action is Delay, or when Type is Submodel. |  |  |
| Type—Specification of a particular resource, or selecting from a pool of resources (i.e., a resource set). | Resource, Set | Resource |
| Resource Name—Name of the resource that will be seized and/or released. Applies only when Type is Resource. | Symbol Name [Resources] | Resource 1 |
| Set Name—Name of the resource set from which a member will be seized and/or released. Applies only when Type is Set. | Symbol Name [Resource Sets] | Set 1 |
| Units to Seize/Release—Number of resource units of a given name or from a given set that will be seized/released. For sets, this value specifies only the number of a selected resource that will be seized/released (based on the resource’s capacity), not the number of members of a set to be seized/released. | Expression Truncated to Integer | 1 |
| Selection Rule—Method of selecting among available resources in a set. Cyclical will cycle through available members (1st member-2nd member-3rd member-1st member-2nd member-3rd member). Random will randomly select a member. Preferred Order will always select the first available member (1st member if available, then 2nd member if available, then 3rd member). Specific Member requires an input attribute value to specify which member of the set (previously saved in the Save Attribute field). Largest Remaining Capacity and Smallest Number Busy are used for resources with multiple capacity. Applies only when Type is Set. | Cyclical, Random, Preferred Order, Specific Member, Largest Remaining Capacity, Smallest Number Busy | Cyclical |
| Save Attribute—Attribute name used to store the index number into the set of the member that is chosen. This attribute can later be referenced with the Specific Member selection rule. Applies only when Selection Rule is other than Specific Member. Does not apply when Selection Rule is Specific Member. If Action is specified as Delay Release, the value specified defines which member (the index number) of the set to be released. If no attribute is specified, the entity will release the member of the set that was last seized. | Symbol Name [Attributes] |  |
| Set Index—The index number into the set of the member requested. Applies only when Selection Rule is Specific Member. If Action is specified as Delay Release, the value specified defines which member (the index number) of the set is to be released. | Expression | 1 |
| Delay Type—Type of distribution or method of specifying the delay parameters. Constant and Expression require single values, while Normal, Uniform and Triangular require several parameters. | Constant, Normal, Triangular, Uniform, Expression | Triangular |
| Units—Time units for delay parameters. | Seconds, Minutes, Hours, Days | Hours |
| Allocation—Determines how the processing time and process costs will be allocated to the entity . The process may be considered to be value added, non-value added, transfer, wait or other and the associated cost will be added to the appropriate category for the entity and process. | Value Added, Non-Value Added, Transfer, Other, Wait | Value Added |
| Minimum—Parameter field for specifying the minimum value for either a Uniform or Triangular distribution. | Expression | .5 |
| Value—Parameter field for specifying the mean for a Normal distribution, the value for a Constant time delay, or the mode for a Triangular distribution. | Expression | 1 |
| Maximum—Parameter field for specifying the maximum value for either a Uniform or Triangular distribution. | Expression | 1.5 |
| Std Dev—Parameter field for specifying the standard deviation for a Normal distribution. | Expression | .2 |
| Expression—Parameter field for specifying an expression whose value is evaluated and used for the processing time delay. | Expression | 1 |
| Report Statistics— Specifies whether or not statistics will automatically be collected and stored in the report database for this process. | Checked, Unchecked | <Checked> |

**0.6.3 DISPOSE MODULE**

This module is intended as the ending point for entities in a simulation model. Entity statistics may be recorded before the entity is disposed.

**0.6.3.1 TYPICAL USES**

* Parts leaving the modeled facility
* The termination of a business process
* Customers departing the store

|  |  |  |
| --- | --- | --- |
| Prompt | Valid Entry | Default |
| Name—Unique module identifier. This name is displayed on the module shape. | Symbol Name [All Modules] | <module name and instance number> |
| Record Entity Statistics—Determines whether or not the incoming entity’s statistics will be recorded. Statistics include value added time, non-value added time, wait time, transfer time, other time, total time, value added cost, non-value added cost, wait cost, transfer cost, other cost, and total cost. | Checked, Unchecked | <Checked> |

**0.6.4 ASSIGN MODULE**

This module is used for assigning new values to variables, entity attributes, entity types, entity pictures, or other system variables. Multiple assignments can be made with a single Assign module.

**0.6.4.1 TYPICAL USES**

* Accumulate the number of subassemblies added to a part
* Change an entity’s type to represent the customer copy of a multi-page form
* Establish a customer’s priority

|  |  |  |
| --- | --- | --- |
| Prompt | Valid Entry | Default |
| Name—Unique module identifier. This name is displayed on the module shape. | Symbol Name [All Modules] | <module name and instance number> |
| Assignments—Specifies the one or more assignments that will be made when an entity executes the module. |  |  |
| Type—Type of assignment to be made. Other can include system variables, such as resource capacity or simulation end time. | Variable, Variable Array (1D), Variable Array (2D), Attribute, Entity Type, Entity Picture, Other | Variable |
| Variable Name—Name of the variable that will be assigned a new value when an entity enters the module. Applies only when Type is Variable, Variable Array (1D), or Variable Array (2D). | Symbol Name [Variables] | Variable 1 |
| Row—Specifies the row index for a variable array. | Expression truncated to a non-zero integer | 1 |
| Column—Specifies the column index for a variable array. | Expression truncated to a non-zero integer | 1 |
| Attribute Name—Name of the entity attribute that will be assigned a new value when the entity enters the module. Applies only when Type is Attribute. | Symbol Name [Attributes] | Attribute 1 |
| Entity Type—New entity type that will be assigned to the entity when the entity enters the module. Applies only when Type is Entity Type. | Symbol Name [Entity Types] | Entity 1 |
| Entity Picture—New entity picture that will be assigned to the entity when the entity enters the module. Applies only when Type is Entity Picture. | Symbol Name [Entity Pictures] | Picture.Report |
| Other—Identifies the special system variable that will be assigned a new value when an entity enters the module. Applies only when Type is Other. | Expression | J |
| New Value—Assignment value of the attribute, variable, or other system variable. Does not apply when Type is Entity Type or Entity Picture. | Expression | 1 |

**0.6.5 DECIDE MODULE**

There are four types of Decide modules:

2-way by Chance

2-way by Condition

N-way by Chance

N-way by Condition

The Decide module's default type is 2-way by Chance. There are two exit areas associated with this module type. Entities that meet the True condition exit from the right side of the module, while entities that meet the false condition exit from the bottom of the module. The exit behavior is the same for Decide modules with type 2-way by Condition.

For a type defined as N-way by Chance or N-way by Condition, the exit point to the right of the module is a repeatable exit point, which consists of an exit for each of the conditions or probabilities specified. The exit point at the bottom of the module is defined to cover any Else conditions.

All exit points must be connected to other modules to run the simulation. Multiple exit points can be connected to the same module, if desired.

When using a Condition type of module, if all specified conditions are false, the entity will automatically exit from the False or Else exit at the bottom of the module. When using a Chance type of module, a random sample is generated when an entity arrives at the module. If the random sample falls outside the percent true ranges specified, the entity will automatically exit from the False or Else exit at the bottom of the module.

When entering values for by Chance decisions, enter the value as a number, not as a percentage. For example, to specify a true condition of 87%, enter 87, not .87. If you are loading variable values from a spreadsheet, make sure the percentage values in the spreadsheet are expressed as whole numbers, not decimal values.

**0.6.5.1 TYPICAL USES**

* Dispatching a faulty part for rework
* Branching accepted vs. rejected checks
* Sending priority customers to a dedicated process

|  |  |  |
| --- | --- | --- |
| Prompt | Valid Entry | Default |
| Name—Unique module identifier. This name is displayed on the module shape. | Symbol Name [All Modules] | <module name and instance number> |
| Type—Indicates whether the decision is based on a condition (if X>Y) or by chance/percentage (60% yes, 40% no). The type can be specified as either 2-way or N-way. 2-way allows for one condition or probability (plus the "false" exit). N-way allows for any number of conditions or probabilities to be specified as well as an "else" exit. | 2-way by Condition, 2-way by Chance. N-way by Condition, N-way by Chance | 2-way by Chance |
| Conditions – Defines one or more conditions used to direct entities to different modules. Applies only when Type is N-way by Condition. |  |  |
| Percentages – Defines one or more percentages used to direct entities to different modules. Applies only when Type is N-way by Chance. |  |  |
| Percent True—Value that will be checked to determine the percentage of entities sent out a given true exit. | Expression (Various percentage alternatives) | 50 |
| If—Type of conditions that are available for evaluation. | Variable, Variable Array (1D), Variable Array (2D), Attribute, Entity Type, Expression | Entity Type |
| Named—Specifies either the name of the variable, attribute, or entity type that will be evaluated when an entity enters the module. Does not apply when Type is Expression. | Symbol Name [Variables, Attributes, Entity Types] | Variable 1, Attribute 1, Entity 1 |
| Row—Specifies the row index for a variable array. | Expression truncated to a non-zero integer | 1 |
| Column—Specifies the column index for a variable array. | Expression truncated to a non-zero integer | 1 |
| Is—Evaluator for the condition. Applies only to Attribute and Variable conditions. | >=, >, ==, <>, <, <= | >= |
| Value—Expression that will be either compared to an attribute or variable, or that will be evaluated as a single expression to determine if it is true or false. Does not apply to Entity Type condition. If Type is Expression, this value must also include the evaluator (e.g., Color<>Red). | Expression | 1 |

**0.6.6 DELAY MODULE**

The Delay module delays an entity by a specified amount of time.

When an entity arrives at a Delay module, the time delay expression is evaluated and the entity remains in the module for the resulting time period. The time is then allocated to the entity’s value added, non-value added, transfer, wait or other time. Associated costs are calculated and allocated as well

**0.6.6.1 TYPICAL USES**

* Processing a check at a bank
* Performing a setup on a machine
* Transferring a document to another department

|  |  |  |
| --- | --- | --- |
| Prompt | Valid Entry | Default |
| Name—Unique module identifier. This name is displayed on the module shape. | Symbol Name [All modules] | <module name and instance number> |
| Allocation—Type of category to which the entity’s incurred delay time and cost will be added. | Value Added, Non-Value Added, Transfer, Wait, Other | Other |
| Delay Time—Determines the value of the delay for the entity. | Expression (Distributions) | 0.0 |
| Units—Time units used for the delay time. | Seconds, Minutes, Hours, Days | Hours |

**0.6.7 STATION MODULE**

The Station module defines a station (or a set of stations) corresponding to a physical or logical location where processing occurs. If the Station module defines a station set, it is effectively defining multiple processing locations.

The station (or each station within the defined set) has a matching Activity Area that is used to report all times and costs accrued by the entities in this station. This Activity Area’s name is the same as the station. If a parent activity area is defined, then it also accrues any times and costs by the entities in this station.

**0.6.7.1 TYPICAL USES**

* Defining a lathe area
* Defining a set of toll booths
* Defining a food preparation area

|  |  |  |
| --- | --- | --- |
| Prompt | Valid Entry | Default |
| Name—Unique module identifier. This name is displayed on the module shape. | Symbol Name [All Modules] | <module name and instance number> |
| Station Type—Determines the type of station, either an individual station or a set of stations. | Station, Set | Station |
| Station Name—Defines the symbol name of the station that is associated with this entrance point. | Symbol Name [Stations] | Station 1 |
| Parent Activity Area—Name of the Activity Area’s parent. | Symbol Name [Activity Area] |  |
| Associated Intersection—Name of the intersection associated with this station in a guided transporter network. | Symbol Name [Intersections] | No associated intersection |
| Report Statistics—Specifies whether or not statistics will automatically be collected and stored in the report database for this station and its corresponding activity area. | Checked, Unchecked | Checked |
| Set Name—Defines the symbol name of the station set that is associated with this entrance point. | Symbol Name [Station Sets] | <module name and instance number>.Set |
| Save Attribute—Specifies the attribute to be used to store the index into the station set for an entity entering this module. For example, assume that a station set has three station members. An entity that is sent to this module is sent to an individual station within the station set. If the entity is sent to the second station in the station set, then this attribute will be assigned a value of 2. This attribute is useful if other sets such as queue or resource sets are defined corresponding to the station set. For example, an entity at the second station in a station set might wait in the second queue in the corresponding queue set and seize the second resource in the corresponding resource set. | Symbol Name [Attributes] | Attribute 1 |
| Station Set Members—This repeat group permits you to define the individual stations that are to be members of the specified station set. A station set must have at least one member station. | | |
| Station Name—The name of a station that is to be a member of this station set. A given station can only exist within a model once. Therefore, an individual station can only be the member of one station set, and that individual station may not be the name of a station in another module. | Symbol Name [Stations] | Station 1 |
| Parent Activity Area—Name of the Activity Area’s parent for the station set member. | Symbol Name [Activity Area] |  |
| Associated Intersection—Name of the intersection associated with this station set in a guided transporter network. | Symbol Name [Intersections] | No associated intersection |
| Report Statistics—Specifies whether or not statistics will automatically be collected and stored in the report database for this station set member and its corresponding activity area. | Checked, Unchecked | Checked |

**0.7 MATHEMATICAL MODEL**

As mentioned earlier, in a parking assignment problem the two main costs associated are individual and social costs. This happens due to the fact that the drivers do not know exactly where is a free parking space available satisfying their own parking criteria’s. So they go on cruising till they can find a place to park.

Again, when a driver approaches a parking lot and finds it filled with vehicles he does not know where to go next. He may try to go to the nearest parking area but there is a very less probability that he will find a free spot there. So if the system decides the parking spot then there is a higher probability that the user will find a vacant space because the system considers the real time status of the nearest parking spot.So basically we have two facts to consider in a parking assignment problem. One is the distance between the approached parking lot and its neighbouring nodes; other one is their space availability.

In all cases, the least distance is not always the best choice. So we cannot assign all the incoming vehicles in a particular node to its nearest neighbour as it will increase the congestion at that particular node. To tackle the distance based problem, can be used as a solution where,

dij: Distance between two neighbouring nodes (Say node Pi and Pj)

Dup: Maximum of the distances between current and the neighbouring nodes.

Now coming to the vacant space availability criteria, again we cannot choose the node with the most vacant spaces because it could be far away from other nodes and hence make the system inefficient. So to solve the resource (here vacant space) assignment problem, a similar approach can be used, such as where,

Sj : Number of vacant spaces available at Pj

Sup: Maximum capacity of the whole network

Both of these above mentioned equations have vital roles in the design of our system. So we have to include both the equations in our system to find an optimal parking spot. To find out the total cost associated with the searching process, consider the equation:

Where C (a, b): total cost associated with the searching process

a: coefficient depending upon the length between two nodes

b: coefficient depending upon the number of vacant spaces in destination node

To find the optimal values of a and b, a simulation was performed using ARENA, which a discrete event simulator tool. For a=0.2, b=0.8 the system shows the most balanced distribution.

**SIMULATION PARAMETERS**

|  |  |  |
| --- | --- | --- |
| **PARAMETER** | **VALUE** | **UNIT** |
| TOTAL NUMBER OF VEHICLES | 30 | VEHICLES |
| INTER - ARRIVAL RATE | POIS(30) | MINUTES |
| SERVICE RATE | EXPO(120) | MINUTES |
| VALUE OF a | 0.2 |  |
| VALUE OF b | 0.8 |  |

0.8 CONCLUSION AND FUTUREWORK

The purpose of this roadmap is to simplify the complexity of parking across parking lots by implementing a smart parking system, which can provide a balanced solution to unbalanced parking demands.

IoT simply means to transmit data of a sensor for electronic equipment through the medium of internet. By using the principle of IoT and sensors we will try to solve the problem of unbalanced parking demands along with that to make the whole parking system a completely automated one.

In the era of smart generation, smart parking is a must and it should be implemented across the globe.