Project Title:

Smart Junction [CS3]

Category:

Computer Science (CS)

Synopsis:

Smart Junction is a system designed to reduce the average waiting time of vehicles waiting and intelligently manage the traffic flow in a traffic junction. With the ability to control traffic flow instantaneously and give high priority to emergency vehicles, Smart Junction system is the best way to manage modern world traffic flow. The simulation software we made for testing our concept shows that this system reduces waiting time significantly over the normal junction. The Smart Junction system is composed of three processing phases, Data acquisition, Information processing, and Signal synchronization. Out of these, the first two plays the major role in optimizing traffic flow. Our algorithm –which is the core part of the system– is built in such a way that it creates a deep understanding of the current situation of all its associated roads and prioritizes the signal control accordingly. The algorithm is also designed in such a way that it has failsafe procedures so that there are almost no chances for operation downtime.

Project Information:

Objectives and introduction:

S M A R T J U N C T I O N - I N T R O D U C T I O N A N D O B J E C T I V E S

> *When I was waiting in a junction for our signal to go green, and also seeing that the road which has green now has no vehicles to be cleared, I started wondering, why are we waiting even though that road has no vehicles to be cleared, isn't this system an inefficient approach to manage traffic at junctions.*

Since then my friend and I started working on a new system to manage traffic efficiently and effectively in a junction, and after much research on this for a long time, we were able to bring out the new concept of **'Smart Junction'**. The main objective of this idea was to **reduce waiting time and manage traffic in the most optimal way**, but gradually while developing this idea, we were able to incorporate many other features which would allow **high priority optimizations for roads having emergency vehicles** such as an ambulance. It will also lead to safe passage of the emergency vehicle as the signal control system would automatically adapt for safety, unlike normal junction signals.

The system understands the number of vehicles *(not traffic density to be precise as it is explained in the algorithm)* and accordingly, the road is assigned a priority/preference value. The road having the highest

priority value will get to clear next, and this was further stretched to incorporate emergency priority, where very high priority values are dedicated when emergency vehicles *(like ambulance, fire and rescue vehicle, etc.)* approaches the junction. The major approach in which waiting time can be reduced significantly is by closing the clearing road earlier than the timeout if there are no vehicles to clear and also, allows few seconds extra beyond timeout if only very few vehicles are there to exit *(conditions apply, detailed info about this in the algorithm)*. The smart junction system adapts to the traffic conditions instantly. There are junctions which adapt to traffic conditions in some countries, but those are pre-configured prioritization for such as during business hours, so since our smart junction adapts instantaneously, it has a very high potential to outperform those and can be the **best solution for managing and optimizing traffic flow in the most effective and efficient way**.

Main Objectives

- **Optimize traffic flow**.
- **Reduce waiting time** of vehicles in the junction *(even by closing the roads before its actual timeout).*
- **High preference for emergency vehicles** (like ambulance, fire and rescue, etc.).
- Do early optimizations for that road so that the emergency vehicles wouldn't be facing traffic congestions and can go at high **speed**, at the same time maintaining **safety** for other vehicles inside the junction box.
- Instantly **adapt** to current traffic conditions.

Extended Objectives

- **Vehicle tracking** *(depends on vehicle detection method, explained in the algorithm section)* for traffic polices or for supporting any other legal purposes.
- **Live traffic information for public** and client services like Google Traffic (as their method of obtaining traffic congestion levels isn't always accurate).
- Large-scale deployment can lead to much better optimization of traffic-flow as junctions would be able to communicate traffic information to their neighbouring junctions.
- As waiting time is reduced*, it can make people happier as they'll have to wait lesser in their daily commute
- Reduces pollution* caused due to engine uptime for no movement.
- A major advantage of this project is that this system can be imposed upon existing junctions and it does not require any major setup modifications. It also does not require any major maintenance.

  *a significant reduction is only observed in a large-scale deployment

Innovation:

THE RICH ALGORITHM DESIGNED TO OPTIMIZE MODERN WORLD TRAFFIC

The most innovative part of our concept is the rich and powerful algorithm which we have designed as it is capable of handling almost all sorts of traffic conditions.

Innovative features of the algorithm:

- **Analyzes traffic conditions, and performs traffic flow optimizations better and better for every iteration**, while at the same time countering road downtimes.
- **Reduces waiting time significantly** and is mathematically designed to achieve the same.
- Traffic flow optimizations are done after **deeply understanding the current traffic condition** in a fraction of seconds *(instant adaption).*
- Has built-in failsafe procedures, to prevent traffic clogging.
- Ability to give **higher preference to roads having emergency vehicles**, and is also capable to handle multiple emergency vehicles coming from different roads, where safety is given higher importance in this situation.
- External control: this algorithm has additional modes of operation. Normal junction mode and Manual mode. In manual mode, the authorities can control the junction as per their requirements.
- Streetlight: The program can also control the street light to minimise energy usage and accidents*. When there are no vehicles in a road the street lights dim (bright enough for people) and as soon as vehicles approach they increase brightness.

  *when streetlights are not bright enough, vehicles shine their headlights in maximum. this causes many problems for other drivers including accidents

Algorithm:

S M A R T J U N C T I O N S Y S T E M W O R K I N G P R O C E D U R E

1. Obtaining Traffic Information

The information required by the system to get the desired results are the number of vehicles on each road and a list of emergency vehicles approaching the junction. There are multiple ways to get this data.

Getting - Number of vehicles on each road:

1. **Camera Image Visualisation** - In the recent years, there's been an exponential growth in AI and object detection from images using a combination of hardware and software. There's even a google's open source API for image visualisation called **TensorFlow** which is trained on the Microsoft's COCO (Common objects in context) dataset, which is capable of detecting vehicles. Using such API's and a live video input from the camera in the junction, this system will be capable of obtaining the number of vehicles on each road. To make things *scalable to a large extent, minimising cost, and improving accuracy*, it's enough if the system wiretaps into a **PNR (plate number recognition) camera**, which already exists in almost every junction, so that this way the PNR camera becomes the embedded entity of the smart junction system and will have more information than using API's like TensorFlow. But, this method of obtaining data, is just for backwards compatibility and for recognizing vehicles which do not have RFID tags.

2. **Vehicle RFID Mapping** - Nowadays, most of the vehicles come with RFID tags sealed on the vehicle's windshield. These RFID tag's main focus was for recognising the vehicle's information in a small area. This can be used for getting the number of vehicles.

Getting - Emergency Vehicles

Emergency vehicles *(ambulance, fire and rescue vehicles, etc)* should be having a module which can communicate with the Smart Junction services server. Once the emergency mode of the vehicle is activated by the driver, the module will beam the positional data and the destination of the emergency vehicle to the server. And then the server traces its path to the destination and analyzes which all Smart Junctions the vehicle will pass through, and relays an **Emergency Priority** signal to all those junctions. Now the junction will be able to give high preference to the road in which the vehicle would be approaching and optimizes its traffic flow before the emergency vehicle approaches, eliminating traffic congestions for the emergency vehicles.

*Emergency priority value of a vehicle goes higher as the distance to the junction decreases and also has a multiplication factor set by the vehicle driver depending on the intensity of the purpose of emergency.

**The software will monitor the activity of the registered emergency vehicle drivers, to make sure that this feature is not exploited and misused.

2. Information Processing (Execution Cycle) Algorithm:

1. It first checks whether the Junction mode is set to **off** for any external reasons and if yes, it terminates the execution cycle after setting all the signal modules to null. *(Signal modules contain memory instructions to communicate with the hardware implementation of the signals.)*
2. The program maintains a timer to know when to perform next activity and for the algorithms to work. So it decrements the timer at this phase for rest of the execution purpose. This timer is also used for analyzing the waiting time of vehicles for statistical purposes.
3. Priority levels *(a value between 0 to 255)* are manipulated based on the data acquired. Lesser the number of vehicles, more the priority value (passively true statement). *Mathematically this would reduce waiting time although it may sound the opposite for some people, because more the number of vehicles on other tracks, the more the vehicles of the track having least number of vehicles would be waiting, but again this would come with a drawback at times which would cause the busy roads to continue stacking up further and further, so to counteract all these flaws,* ***our priority level calculation formula is crafted in such a way that the local minimum of the 'priority level' vs 'no. of vehicles' graph is at when the number of vehicles hits a threshold value*** *(this value varies over time and the minor variations are manipulated by the algorithm based on the analysis of past traffic optimizations results so that the traffic is optimized even better for the next run every iteration. Also, the value is restricted to a certain range so that large variations caused due to temporary road downtime for any external reasons is prevented too.)*
4. If there are any emergency vehicles, it overrides the rest of the execution and opens the road where the emergency vehicle is approaching for clearance. To make this failsafe, if multiple emergency vehicles are approaching from different roads, then the emergency vehicle having the highest priority value will be given more preference.

(This emergency feature becomes quite important, but taking into real-world factors like in an extremely busy junction, and if the system fails to completely optimize the traffic flow, it still blocks other roads from clearing, thus eliminating possible accidents which could happen in normal junctions)

5. If no emergency vehicles are there, it performs the smart algorithm. If there are no vehicles in the currently open road, then it closes the road and gives the next road the chance to clear *(execution jumps to 7)*. This is the main feature which reduces the waiting time. *Just like what a traffic police would do. As observed, even in normal junctions with a fair amount of traffic, all the vehicle get cleared before the fixed time.*
6. If the time left for the current road to close is 0 seconds, and if only very few amounts of vehicles are there left to exit, the time left is incremented by few seconds extra. But the algorithm makes sure that this extra time is not given to a road more than once. *(Just like what a traffic police would do).*
7. Once the road has closed, it waits for a short period of time *(Road turn over delay)* and then opens the road which has the highest priority value. S*ince, lower the number of vehicles, higher the priority value as per the algorithm design, in practice,if the same road that was closed before has the least amount of vehicles, it shouldn't be opening the same road again, for that the algorithm keeps track of what roads have been given a chance in the 'recent_pasts' memory so that every road gets at least one chance before the 'recent_pasts' memory is cleared*.

**Now this algorithm automatically acquires a feature which is not explicitly programmed, that if the number of vehicles on the road is 0 (for any reason) the priority value is not calculated and so its default value is 0, so would skip execution for that road, also if in case that fails to skip, the Early Exit feature automatically kicks in and skips the road.*

3. Signal (Software-Hardware) Synchronization

The above execution cycle deals only with the software definition of roads and does not make changes to the signal modules. Signal synchronization ensures that the system is failsafe in every way. Also if the junction mode is set to manual control *(for any reason)* this module ensures that only one road is set to green signal, and overrides even if the software memory has multiple roads kept open. This also facilitates greater security in case the smart junction system gets breached/hacked. The hardware interface reads the signal states of each road from this module.

The processing phases are separated into three, for easy modifications to be made to the software, and also to make these three isolated from each other from computing time, as 'Obtaining Road Information' updates the road information memory at a different rate than that of the execution cycle.* *(this way, even if the data obtaining fails, the execution algorithm detects that the memory hasn't been updated for quite a time, and understands something is faulty, so it immediately shifts to loop mode)***

**The engine library (core program files) needs to be embedded into a platform-specific software, this is so that all the junctions will have the same operation principle even on different platforms. To test this, we have built a simulation software which is further detailed in the 'Method' section.*

Method:

S M A R T J U N C T I O N - S I M U L A T I O N S O F T W A R E

Since this Smart Junction is a conceptual idea, we had to extensively test it, to understand till what extent it would perform better than a normal junction and what all modifications should be made to our concept. So we built a simulation software, in which the Smart Junction's core engine (with some additional features for geo-specific modifications) is embedded. Now as mentioned earlier in one of the failsafe procedures, the Smart Junction has also got an optional loop mode available in the 'operation modes' options, and we'll be using mode to simulate the currently existing normal junction for getting a comparison between the two.

So for the simulation software to give true results and for greater extent in re-modification, we built all the simulation software's definition files ourselves *(instead of using open source libraries from outside)* just so that we know that the observations made are accurate and has no/minimal margin for errors.

High-Fidelity Simulation *(Real-life imitation of an object's behaviour)*

To make sure that the observations made from the simulation are completely relatable to the practical world, we defined every object in the software extensively to imitate its true behaviour, so as to be classified under **High-Fidelity Simulation**. For examples such as the slight latency caused by the driver when the signal becomes green, slowing down of the vehicle while turning, and many other such small factors. Also, no two vehicles in the simulation will have identical behaviour. Every time a vehicle is to be spawned into the simulation scene, it gets a random vehicle from a pre-configured list of vehicles having different performances and also is assigned with a random driver which is again from a pre-configured list of drivers having different driving preferences.

Simulation Methodology

The simulator is capable of **spawning vehicles at random** *(manual spawning also available for controlled simulation)* and the probability of spawning can be adjusted to simulate different traffic levels. Once the vehicle is spawned, the vehicle starts to drive towards the junction and exits once it's green. Each vehicle has got it's dedicated analytical modules to measure how much time was spent waiting, and many other factors which are important. (*The simulator is designed not to spawn vehicles when there is already another vehicle in the spawn location (which could happen in extremely busy traffic settings), so as to prevent vehicle collisions as these vehicle objects are already programmed to prevent collisions with each other)*

Now as the first phase of the algorithm is *obtaining the traffic levels of road*, the simulation software itself measures the number of vehicles on each road, to a particular distance from the junction *(as all vehicle detection sensors are limited to a range)* and this data is directly fed into the *core algorithms* memory and executed, and the results generated *(signals state)* by the Smart Junction's core algorithm is fed back to the simulation scene. **The simulation software also analyzes the traffic congestion levels of each road over time and keeps a record of events and analysis in a log file for reviewing later.**

For simulating on how the Smart Junction would react to **emergency vehicles**, few additional vehicles and drivers are preconfigured for high speed, high acceleration driving. These emergency vehicles can be spawned manually or randomly *(very low probability of spawning).*

Now for making a **true comparison between both normal junction and our smart junction**, our simulation software is designed to run multiple simulation scenes simultaneously, and so both the types of junctions are loaded, and the simulation software spawns vehicles in both the junctions at the same time equally so as to get a valid ideal comparison in fewer simulation iterations. *(We had also noticed when simulating these two junctions with a quite high vehicle inflow setting, the simulator does not spawn vehicles in the normal junction when the road is the, because as said earlier to prevent vehicle collision, and since this would lead to inaccurate comparison results, the simulator has been updated to not spawn vehicles when spawning would fail in either of the junction.)*

The simulation software is designed to run at higher speeds than real-world time also, so as to generate log files and more accurate analysis (as analysis data becomes more precise over an extended period of simulation). And we have run the simulation for both the junctions for different traffic levels too and was able to create above thirty-seven days (simulation world time) worth analysis and log files after running it for different traffic levels in thirty-six hours (real-world time).

Conclusions/Results:

S I M U L A T I O N R E S U L T S A N D C O N C L U S I O N S

After generating about thirty-seven days worth simulated data and vehicle's wait duration analysis for both smart junction and normal junction, the **Smart Junction performs exponentially better than normal junction** in most of the traffic conditions.

Junction configuration used for simulation:

4 Road / Cross Road Junction; Normal Junction timeout duration: **30 seconds**

<u>AVERAGE</u> WAIT DURATION OF VEHICLES:

At usual/average daily traffic conditions:

Normal Junction: **38.7 seconds** Smart Junction: **21.6 seconds**

At non-busy days / fair traffic conditions:

Normal Junction: **34.1 seconds** Smart Junction: **13.2 seconds** *(significantly reduced as **early timeout feature** kicks in more frequently and earlier)*

At busy traffic conditions:

Normal Junction: **46.2 seconds** Smart Junction: **37.4 seconds** *(still performs better than normal junction as **extra time for very few vehicles to exit feature** kicks in more often)*

We know that these waiting time differences are not really significant. That is because these are the data collected from a single junction. When this concept is applied to a lot of junctions the total integrated value of average waiting time is significantly better than normal junctions

**although the normal junctions timeout duration is thirty-seconds, the average waiting time of vehicles is not a constant ninety seconds as per our junction configuration, because it calculates the waiting time experienced by all the vehicles on each and not the red signal duration as to get a meaningful analysis.*

Conclusion

We are very confident that this concept of Smart Junction is the best way to manage and optimize exponential growth of modern world traffic flow in the most effective and efficient way, as it incorporates solutions for almost all problems faced in normal junctions.

Acknowledgement and reference links:

Online References:

Vehicle Detection Sensors - More ways of vehicle detection.

Assist credits:

Special thanks to Chinmaya Vidyalaya, our beloved teachers, friends and all those who extended their support for this project.

We thank The Chinmaya Mission for supporting us throughout.

We wish to also thank

- Google, for helping us provide the most favourable solutions while researching on this.
- Stackoverflow.com and Microsoft C# Program Documentation for providing us support with .NET frameworks and C# language's built-in definitions. It helped a lot to learn the powerful and advanced data management capabilities of C#.

Have you participated in any Science Completions/ Fair/Exhibitions in the last one year?

CBSE Science Exhibition