Using Decision Support Systems for Transportation Planning Efficiency

Ebru V. Ocalir–Akunal
Gazi University, Turkey
Chapter 10

Sustainable Transport System: Transport On-Demand

Maria Spichkova
RMIT University, Australia

Margaret Hamilton
RMIT University, Australia

ABSTRACT

Transport systems are major emitters of greenhouse gases, which makes environmental sustainability of any transport a crucial issue. Another issue is the lack of a systematic approach to the modeling and implementation of public transport systems. Finally, there are problems with the human interfaces to public transport systems, which do not encourage, and many do not allow, comfortable and simple interaction with the system. In this chapter, the authors discuss their solutions for these problems, explaining how to cover the existing gaps in a methodological and systematic way. The main contribution of this chapter is a model of an on-demand transport system that covers all the points mentioned above and focuses on spatial planning and optimizations including environmental issues in transport planning.

INTRODUCTION

One of the powerful drivers for sustainable transportation is public transport. The more people who share the one means of transport, the less the cost and damage to the environment. However, there are many specific problems and challenges, which make the use of public transport inaccessible and inconvenient for potential new passengers. Some of these issues, which we will consider in this chapter, are:

- **Time** - Long waiting times. A passenger may be required to wait indefinitely for the next bus/train/tram, and any time longer than 20 minutes is perceived as being too long;
- **Distance** - Long walking distances to the next nearest stop/station. Passengers may live a long way away from the nearest public transport route, or the connecting multimodal options may be far separated;

DOI: 10.4018/978-1-4666-8648-9.ch010
Sustainable Transport System

- **Timetables** - Non-punctuality. If the public transport vehicle is not following the timetable, e.g., is running late due to disruption, delays, congestion, passengers may miss their connections and be late for work.

It might appear that increasing the number of services and so making the timetable denser could solve these problems, but this solution would mean

- higher costs for the transport company, and
- further emission of greenhouse gases.

Another factor which could make this solution even more questionable would be if the bus were to be empty (idle drive) which could happen for some parts of any route during some times of the day. Hence quick and obvious options are not always the best solutions, and a systematic approach is needed to deal with these problems. A crucial question here also is, how to model this idea in such a way that it

- is applicable to any kind of transport system; and
- allows extensions for the special cases of transport systems having different flexibility in their operation possibilities; and
- covers modelling of exceptions (problems with the traffic due to some disruptions).

In this book chapter we will develop and explain our model for making public transport sustainable, transport on-demand, through the use of various definitions, examples and scenarios. Our methodology is aimed at providing smart solutions for sustainable outcomes, so that a transport system becomes a part of a ‘smart city’, cf. (Ercoskun, 2011; Nam & Pardo, 2011). We conform with the desire to develop sustainable software and systems, i.e., such software and systems that “meet the needs of the present without compromising the ability of future generations to meet their own needs” (World Commission on Environment and Development, 1987). The sustainability of software and systems development becomes increasingly important every year for many reasons, including expanding the software usage as well as growing demand for building further applications based on it. The three pillars of sustainability, namely the economy, society, and environment need are not balanced, and to deal this problem, a “green software” solution is required, cf. also (Hilty & Aebischer, 2014; Easterbrook, 2010). We also work within the classification of green software according to the Green Software Engineering project where the aim is to apply “green” principles to “software products, software development processes and their underlying software process models”, (GreenSoft, 2009). However, we mainly focus not on the green development process, but on the development of software and systems that can have a positive impact on the environment and human beings.

The main idea behind our approach is to identify whether a bus, tram, or train would normally be empty during the day/night for some parts of particular routes. It could be empty for some cases if the timetable were denser or if the route were extended to outlying districts. We aim to have these parts of the route provided as optional choices in the timetable, during these hours. This would imply both lower costs for the transport company with less emission of greenhouse gases, as well as enabling the transport system to be more attractive for actual and potential passengers due its flexibility. To make this idea realistic, a proper model of the system is essential.
We present our vision of a decision support system for the drivers of public transport. Transport on-demand means dynamic timetables, which are influenced by passengers in real time. For this case drivers need additional assistance from the system to avoid manual analysis to decide which parts of the route should be served/skipped in each particular case. Thus, a decision support system (DSS) should be a key component of the on-demand transport system, to ensure human-oriented development of the driver interface to the system and as a result safety on the road.

The rest of the chapter is organised as follows: In the next section, background information on the related research is given. Then our model for transport on-demand and the corresponding optimisation methods is introduced. The formal model description and the presentation of the system architecture are followed by two on-demand bus scenarios, for a simple general example and for the case study, which is based on the spatial information of an existing real bus route 445 operating in the Melbourne area of Victoria, in Australia. The case study includes a prototype of the system, implemented as a web application. After this, the future research directions are discussed. Finally, the main contributions of the presented work are evaluated in the conclusion section.

BACKGROUND

Our aim is to design a sustainable on-demand transport system model, which is good for passengers and drivers. Unfortunately, we know that the most unreliable component of any computer system is the people who use it. This idea, applied to formal methods, is termed the Engineering Error Paradigm (Redmill & Rajan, 1997). Human factors targeted by the engineering error paradigm typically include the design of the user interfaces as well as any corresponding automatization. In this paradigm people are seen as being equivalent to software and hardware components in the sense of operations with data and other components, but at the same time humans are seen as being “the most unreliable component” of the whole system because they are inconsistent and unreliable. This implies that designing humans out of the main system actions through automatization of some system design steps is considered a good proposal for reducing risk. A decision support system (DSS) can be viewed as a special way of partially designing humans out of the main system actions, particularly if we assume that the human will follow the decision recommended by the support system, cf. e.g., (Marakas, 2003; Bonczek et al., 1981; Fick & Sprague, 1980). DSSs have been designed for industrial and organisational users, but as web-based technologies have progressed, the applications area has become more broad (Bhargava et al., 2007). Large parts of DSSs are now developed for casual users, e.g., for passengers interested in taking the fastest connection from some address to a desired destination point. Moreover, from the development of traditional “desktop applications based” DSSs have evolved to the development of distributed, web-based applications (Power, 2013; Mitra & Valente, 2007). In this chapter we propose a web-based DSS for the public transport driver, where the request information for the route is collected from the passengers dynamically in realtime.

We recognise that another problematic area for transport on-demand systems is the user interface for these systems, both for passengers and for drivers. The research field of Human Computer Interaction (HCI) is concerned with the development of interfaces for every-day applications, and there are many approaches to integrating human interface engineering with software engineering, e.g., (Volpert, 1991; Heumann, 2002; Constantine & Lockwood, 2002; Constantine & Lockwood, 1999). Many early approaches in this area were initiated because of mistakes in usage and the development of safety-critical
One of the widely cited examples of the HCI-related accidents in safety-critical systems is the case of massive radiation overdoses of the Therac-25 machine used in treating cancer over 1985-1987. The machine, produced by Atomic Energy of Canada Limited (AECL), delivered overdoses leading to deaths and serious injuries of patients, some of whom were given over one hundred times the normal dose of radiation (Miller, 1987; Leveson & Turner, 1993). The causes of these accidents have been attributed to software failures as well as to problems with the system interface and the lack of decision support for the operators of this machine. Operators of the Therac-25 were not able to recognise dangerous error situations and continued the treatment even after the machine showed warning messages. These types of accidents have shown that studying the human errors and their causation should be a significant part of software and system engineering at least in the case of safety-critical systems.

An appropriate system interface allowing correct human computer interaction as well as decision support is equally as important as the correct, error-free behaviour of the developed system. Even if the system we develop behaves in the correct way, this would not help much for a case where the system interface is unclear to the user or is too complicated to be used correctly. According to statistics presented in (Dhillon, 2004), the human is responsible for 30% to 60% of the total errors which directly or indirectly lead to the accidents, and in the case of aviation and traffic accidents, 80% to 90% of the errors can be attributed to human error. Thus, it is essential to take human factors into account when developing computer systems, especially decision support and safety-critical systems.

The fundamental goal of human factor engineering, as claimed in (Wickens & Hollands, 2000), is to reduce errors, and increase productivity and safety when the human interacts with a system. Engineering psychology applies psychological perspectives to the problems of system design and focuses on the information-processing capacities of humans. The goals of modelling (including formal methods) are almost the same: to reduce errors, and increase the productivity and safety of the developed systems.

Another important point of the engineering error paradigm is that human errors often occur as a result of mismatches in the HCI and overestimation of the physical capabilities of a person or of some devices. A decision support system can provide the required estimation automatically, consistently and accurately using predefined schemas.

There are several applications of formal methods to HCI analysis, e.g., (Shackel & Richardson, 1991; Følstad et al., 2012), as well as many approaches to system modelling, e.g., (Broy et al., 2008; Berger, 2002; Donath et al., 2008). In our previous work (references have been removed for double-blind review), we focused on human factors in modelling processes as well on readability and usability of formal languages and interfaces, with the aim of combining the achievements in the fields of HCI and modelling approaches.

In our earlier research on the Melbourne public transport system, we identified that 90% of commuters consider the time spent travelling as their “in-between” time and use it to catch up with their family and work colleagues on mobile phone or email (references have been removed for double-blind review). This demonstrates a willingness of the majority of commuters to connect, and indicates there could be a willingness to engage with the public transport system to inform their journey, if such an opportunity were available. Also, earlier work on an automatic passenger counting system installed on Yarra Trams (tram network in Melbourne, Australia), highlighted that if commuters knew the numbers of other people travelling on their route in advance, many would consider alternative options (Fihn & Finndahl, 2011).

The approach we present in this chapter is based on the experiences from several projects of software and systems engineering in mobility, modelling, formalization and testing, especially within existing...
transport systems and automotive fields in Europe and Australia, cf. e.g., (Hamilton & Berry, 2010; Hamilton et al., 2011; Spichkova et al., 2012; Spichkova, 2011; Feilikas et al., 2011). In the rest of the chapter, we will emphasize the development of a formal model for incorporating the various personalized transport solutions for on-demand public transportation planning systems.

DEVELOPING A MODEL FOR TRANSPORT ON-DEMAND

In this section we introduce the idea of load modes for the transport system, based on passenger numbers at particular places for various times. We introduce our model and develop some corresponding optimizations, which will save not only on fuel costs, but also reduce greenhouse gas emission and noise pollution.

In this chapter, we define a vehicle to be a public transport vehicle, such as a bus, a tram, or a train. The general ideas of our approach can be applicable to any kind of public transport vehicle, including ferries and cars or taxis, but it is easiest to see that bus transport systems can benefit the most from our innovation, due to the flexibility of this kind of vehicle. Bus delays are less critical than for the rest of the transport system because buses can easily deviate from their route, if necessary. This gives us an opportunity for further extensions of the system. For example, the bus stops on the optional parts of the route can be defined as “floating” stops, so that the bus driver can pick up passengers between the regular stops (assuming that the vehicle stopping is allowed on these streets) thus increasing the overall quality of the service. Another example could be deviations from the route, for hours out of the peak times, to pick up disabled passengers with restricted physical abilities.

We model a public transport route \( R \) from place \( A \) to place \( B \) by a sequence of stops/stations \( [S_0, \ldots, S_n] \). Thus, the set of route stops is defined by \( \text{Stops} = \{S_0, \ldots, S_n\} \), where each stop/station \( S_i \) has special attributes:

1. \( \text{depart}_{T_i} \) – time, when the vehicle should depart on the stop/station,
2. \( \text{location}_{i} \) – spatial information about the stop/station (\( \text{location}_{0} \) corresponds to \( A \), and \( \text{location}_{n} \) corresponds to \( B \)),
3. \( \text{option}_i \) – Boolean marker whether the vehicle should operate on this stop/station of the route (\( \text{option}_i = \text{true} \) means that the vehicle should stop at \( S_i \)).

While modeling a simple route, which operates static timetables and does not allow any additional requests from passengers or dynamic optimization of the route, it is enough to use the first two attributes, \( \text{arrival}_{T} \) and \( \text{location} \). For modelling an on-demand transport system, the option attribute is essential.

In general, we can say that a public transport system has two modes of operation on any route:

- \( \text{irregularLoad} \) mode, when the vehicle could be empty. This means there could be times when no passengers are around to travel over parts of the route, and
- \( \text{normalLoad} \) mode, when there is at least one passenger in the vehicle.

Our goal is to avoid the situations where the bus has an idle trip for all or parts of its route. In the ideal case, the system always runs in the \( \text{normalLoad} \) mode, but if not, we can recommend a system optimization in the following manner:
Case 1. End of route. When the irregular connections are at either end of the route, the optimization is implemented in the real transport system in a straightforward way discussed below, in the next section.

Case 2. Along route. Optimization for the irregular connections in the middle of the route (i.e., anywhere between “regular” connections) is practical for the cases when the bus can take a shortcut along the route, taking a shorter trip to the beginning of the “regular” connections.

We call the timetable for the optimized parts of the route optional, because the vehicle follows this timetable and approaches the corresponding stops/stations only on demand, when requested by a passenger.

For each particular run of the route $R$, we also define $\text{RegularStops}^R$ to be the sequence of stops/stations, on which the vehicle should stop during this run, i.e. for all $i$, $0 \leq i \leq n$,

$$S_i \in \text{RegularStops}^R \Leftrightarrow \text{option}_i = \text{true}$$

This sequence is statically specified for the normalLoad mode, but in the case of the irregularLoad mode it should be specified dynamically, according to the requests of passengers.

**Route Optimisation**

**End of Route Optimisation**

For Case 1, when the irregular connections are at either end of the route, these parts of the route become optional (the option attribute for the corresponding stops is set to false), and we introduce so-called provisional ends of the route (stops/stations, where the regular connections end) if

1a. at the provisional end there are no passengers left in the vehicle, and
1b. the driver has no message that the vehicle is requested between the provisional and the actual end of the line (in either direction).

In this case the vehicle need not go to the actual end of the line and can stay at the provisional end until according to the timetable it would need to proceed in the opposite direction. Thus, the driver’s decision, about whether to stop at the provisional end or continue to the actual end, leads to an automatic system operating on passengers’ requests and also on historical statistical data collected about each route.

**Along Route Optimisation**

For Case 2, when the irregular connections occur during the route and not at either end. If there is a possible shortcut, we introduce so-called provisional shortcut stops of the route if

2a. the vehicle can shortcut to miss at least a few of the irregular connections:
   i. the stops/stations between which the shortcut can be taken and become the provisional shortcut ends, and
   ii. the part of the route to shortcut becomes the optional part of the route (the option attribute for the corresponding stops is set to false), and
2b. at the provisional shortcut end there are no passengers left in the vehicle, and
2c. the driver has no message that the vehicle is requested at any stops in the optional part of the route.

In both cases, the request for the vehicle should be sent before some predefined time (which we call the on-demand request time, denoted by $\text{requestT}$) before the vehicle arrives at the stop/station after which it will proceed according to the optional timetable. For usability reasons, we recommend setting this time to the maximum of all required request times within the route, rounding this maximum upward to 5 minutes steps. For example, if the calculated maximum over all required request times were 12 minutes, we would set the on-demand request time to 15 minutes to make it easier for passengers to remember. We demonstrate this in the bus scenario below.

The on-demand request time could also be set not for a single route but for the whole transport system, which would make it easier to remember. However, if the differences in between on-demand request times of several routes exceed 5-10 minutes, it may be more convenient for passengers to have different request times for different bus routes, where for some routes these times are much shorter.

Each optional stop/station $S_j$ obtains an additional parameter, $\text{statistics}_j$, to collect information about how often and when the system has obtained requests to serve this stop/station.

**Architecture of the System**

To have a sustainable and human-oriented solution, we need three views on the transport system (cf. Figure 1 for the architecture of the system):

*Figure 1. Architecture of the on-demand transport system*
Sustainable Transport System

• a **system view**, which comprises the complete information about the route and its runs, and also includes the statistics of the requests;

• a **passenger view**, which is focused on the possibilities of requesting the vehicle to travel to a certain stop/station. It would make no sense for a passenger to send a request for a vehicle to service a particular stop at the time $t$, if there were another earlier request for a vehicle to serve this same stop at this same time. Thus, before sending their own request, passengers need to know the current state of the route – which stops/stations are still optional and which are already “regular” due some previous request(s);

• a **driver view**, which contains only the information that is currently required to complete the run in the most efficient and safest way – the sequence $\text{RegularStops}^s$, where the beginning of the sequence is truncated before the current stop. When the driver leaves a stop, it disappears from his plan (but not from the system view) on this route. The driver view can be represented
  - in a **general mode**, containing the complete current route plan, or
  - in a **simplified mode**, containing the information about 4 stops/stations only: the current stop, which the vehicle is/should be approaching right now, and three next consecutive stops.

In the next sections we present a discussion and comparison of these modes. The main advantage of the simplified mode is better readability of the plan: if there are no disruptions or problems with traffic along the route, it is not important for drivers to know what the distant stops will be or the route actualisation for them, at this point in time. Moreover, dynamically updating information about these stops can distract from safe driving, where a compact view in a simplified mode provides only the information which is currently required. The driver and the system views comprise information on

• the $\text{depart}T_i$ values (i.e. times when the vehicle should depart the stop/station) for each stop/station along the route plan for this run, as well as

• the actual and planned locations (in relation to the stops/stations) of the vehicle at the current moment in time.

In this chapter, we focus on the driver and the system views, as they are important for the decision support system.

**On-Demand Bus Scenario: General View**

Assume an abstract simple bus route consists of 15 stops, cf. Figure 2(a). For this case, we specify a set of $\text{Stops} = \{S_0, \ldots, S_{14}\}$, and represent the route by the sequence $[S_0, \ldots, S_{14}]$. The timetable density of this route is assumed to be reasonable for all hours of the bus operation, and well synchronized with other routes.

For the connections between $S_1$ and $S_5$ as well as between $S_7$ and $S_{12}$, the bus is never empty (denoted by the solid lines in Figure 2), but at the connections between $S_0,$ $S_{14}$, $S_5$, and $S_9$, as well as between $S_{12}$ and $S_{14}$ for some hours $H$, e.g., very early/late hours or middle of the day, the bus is often empty (denoted by dashed lines). We can say that the bus system has two modes of operation on this route, $\text{normalLoad}$ mode (out of the hours $H$) and $\text{irregularLoad}$ mode (during the hours $H$).

In addition, we assume that for the bus travelling along this route, a direct shortcut between stops $S_5$ and $S_9$ is impossible from either direction, and:
Sustainable Transport System

Figure 2. Example of a simple transport route. (a) Identification of the optional parts; (b) Optimization of the route; (c) Route plan at the beginning of the run. We assume there were no requests to stop at $S_0$, $S_7$, $S_8$, $S_{13}$, and $S_{14}$. We also assume that the vehicle waited too long at stop $S_1$ and according to the system plan should be at $S_3$. Hence the actual and the planned locations of the vehicle are different on the plan. (d) Route plan for the Case 1.1; (e) Route plan for the Case 1.2.

- for the direction $S_0 \rightarrow S_{14}$, after the stop $S_5$, the only efficient way for the bus to go is to travel directly to the stop $S_6$. However the bus can shortcut the path $S_6 - S_8 - S_9$ by travelling from $S_6$ directly to $S_9$.
- for the reverse direction $S_{14} \rightarrow S_0$, after the stop $S_9$, the bus can shortcut the route to the stop $S_6$, but cannot go directly from $S_9$ to $S_5$. This could happen, e.g., in the case of a one-way street.

This would mean that the provisional shortcut stops become not $S_0$ and $S_9$, but $S_6$ and $S_9$, see Figure 2(b).

Therefore, we optimize the route by dividing it into 5 parts: two regular parts between $S_1$ and $S_6$ as well as between $S_9$ and $S_{12}$ (the corresponding connections are denoted by solid lines), and three optional...
parts between $S_0$ and $S_1$, $S_6$ and $S_9$, and between $S_{12}$ and $S_{14}$ (denoted by dashed lines). The dotted lines in Figure 2 denote possible shortcuts of the route. $S_1$ and $S_{12}$ become *provisional ends* of the route; $S_6$ and $S_9$ become *provisional shortcut stops*.

The required request time for the optional parts of the route can be calculated as follows:

- The part $S_0 - S_1$: this optional part of the route consists of a single stop only. If, according to the optional timetable, the bus should take $t_{1A}$ minutes to go from stop $S_1$ to stop $S_0$ and $t_{1B}$ minutes to go from $S_0$ to $S_1$, then we can define $t_1 = \text{maximum}(t_{1A}, t_{1B})$ as the maximum request time required for this optional part of the route.

- The part $S_6 - S_9$: When moving from $S_6$ to $S_9$, the bus should take $t_{2A}$ minutes to go from the stop $S_6$ to the stop $S_9$ and $t_{2B}$ minutes to go from the stop $S_9$ to the stop $S_6$, then $t_2 = \text{maximum}(t_{2A}, t_{2B})$ becomes the maximum required request time for this optional part of the route.

- In the same way we can specify the maximum required request time $t_3$ for the part $S_{12} - S_{14}$.

Thus, the *on-demand request time* for this route should be the maximum of $t_1$, $t_2$, and $t_3$, rounded up to the nearest 5 minutes.

In the above scenario, the bus operates on parts of the route only on demand, which increases both profit for the transport company and overall sustainability of the public transport system, without any disadvantages for the passengers. To implement this optimization we need to identify which parts of the route are “optional” and during which hours, and provide an interface to enable comfortable and easy requesting of the bus at the optional stops according to the timetable. As technology improves, the numbers of passengers on board vehicles is becoming more accurately calculated and available. Yarra Trams (providers of the tram network in Melbourne, Australia) have been trialling automatic passenger counting systems, which count the number of passengers who board and exit the tram at each stop (Fihn & Finndahl, 2011). This information can inform the transport providers so they know which stops along the route to identify as optional, and at which times of the day.

In general, the optional section of the route is quite long and has many provisional shortcut stops, because we could have many possibilities for optimizing the route according to the concrete route load and passengers’ requests (an example of such a route is presented in the next section). The choices regarding where to drive and which shortcut to choose should be made very quickly to avoid additional delays in the traffic. This means tighter constraints on the driver if he/she needs to do it manually, using a map, or from memory. Thus, on the one hand this optimization can help to save the environment and make public transport more sustainable in the environmental sense, but on the other hand, if we do not have a decision support system for the driver, the decision overload (caused by the optimization) could lead to human error: driving mistakes on the road, wrong shortcut selection, lack of alertness because of stress and tiredness, not seeing passengers, etc. One of the most common causes of human error in using safety-critical systems is an overestimation of human ability to stay alert for long periods of time or to make decisions in parallel with performing some safety-critical action (as in the case of driving on the road). Following the *Engineering Error Paradigm* (Redmill & Rajan, 1997), we recommend taking the optimisation decision function from the driver to make the system more reliable and road-safe: we suggest to accompany the optimization of the public transport routes with an appropriate decision support system, which can also be seen as a dynamically extended/shortened route during the times when the route is in the *irregularLoad* mode.
Let us discuss the dynamic plan for the route shown in Figure 2 (assuming that the route is in the irregularLoad mode, and while the vehicle goes in the direction $S_0 - S_{14}$). We represent this as a general version using notation $S_0, \ldots, S_{14}$ and listing all the stops along one line, where in the implementation we would be employing a vertical listing (more readable for the small screens) and using the names of the stops as specified in the timetable available for passengers. The stops where some route planning extensions can occur (i.e. the provisional ends and shortcut stops) are marked bold and use orange colour to highlight and draw attention to the possible changes in the connections. In the Case 1, we assume the vehicle was empty at the stop $S_1$ when travelling previously in the opposite direction and there were no requests from the stop $S_0$, before starting to drive in the direction $S_0 - S_{14}$, so the route plan would start from $S_1$. We have six options for updating the current plan of the route, depending on further requests:

**Case 1.** There were no requests from the stops $S_7$, $S_8$, $S_{13}$, and $S_{14}$: The sequence $\text{RegularStops}^k$ for this case is $[S_1, S_2, S_3, S_4, S_5, S_6, S_9, S_{10}, S_{11}, S_{12}]$, and the route plan is defined as follows (cf. also Figure 2(c), which shows the system view as well as general and simplified driver views for this scenario):

$$S_1 - S_2 - S_3 - S_4 - S_5 - S_6 - S_9 - S_{10} - S_{11} - S_{12}$$

In this case, when leaving $S_1$, once more we have six options for the updating of the current plan of the route, depending on the requests, cf. Figure 2(d). We also assume that the vehicle waited too long at $S_1$ and according to the system plan should already be at $S_3$, meaning that the actual and the planned locations of the vehicle are different for this case, as shown in Figure 2(c). Please note, that it is impossible to receive a request for this stop on this run, because this stop already belongs to the “traversed” part of the route. The important point here is that the simplified driver view is the same for all six subcases cf. Figure 2(d3).

**Case 1.1.** There were no requests from the stops $S_{13}$, $S_{14}$, and $S_{15}$, and $S_{16}$: The sequence $\text{RegularStops}^k$ remains unchanged, and the current route is defined as follows, cf. Figure 2(d):

$$S_2 - S_3 - S_4 - S_5 - S_6 - S_9 - S_{10} - S_{11} - S_{12}$$

**Case 1.2.** There was a request from at least one of the stops $S_7$ and $S_8$ (a request from one of this stops makes both of them “regular” for this run), but there were no request from $S_{13}$ and $S_{14}$: The sequence $\text{RegularStops}^k$ becomes $[S_1, S_2, S_3, S_4, S_5, S_6, S_9, S_{10}, S_{11}, S_{12}]$, and the current route is updated, cf. Figure 2(e):

$$S_2 - S_3 - S_4 - S_5 - S_6 - S_9 - S_{10} - S_{11} - S_{12}$$

**Case 1.3.** There was a request from the stop $S_{13}$, but there were no request from $S_{16}$, and $S_{13}$, and $S_{14}$: The sequence $\text{RegularStops}^k$ becomes $[S_1, S_2, S_3, S_4, S_5, S_6, S_9, S_{10}, S_{11}, S_{12}, S_{13}]$, the current route is updated as shown in Figure 3(a):

$$S_2 - S_3 - S_4 - S_5 - S_6 - S_9 - S_{10} - S_{11} - S_{12} - S_{13}$$

**Case 1.4.** There was a request from the stop $S_{14}$, but there were no request from $S_{15}$ and $S_{16}$: The sequence $\text{RegularStops}^k$ becomes $[S_1, S_2, S_3, S_4, S_5, S_6, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}]$, the current route is updated as shown in Figure 3(b):
Figure 3. Route plan for the Cases 1.3 – 1.6

(a) Route plan for the Case 1.3
(a1) System view:
(a2) Driver view:

(b) Route plan for the Case 1.4
(b1) System view:
(b2) Driver view:

(c) Route plan for the Case 1.5
(c1) System view:
(c2) Driver view:

(d) Route plan for the Case 1.6
(d1) System view:
(d2) Driver view:

\[ S_2 - S_3 - S_4 - S_5 - S_6 - S_9 - S_{10} - S_{11} - S_{12} - S_{13} - S_{14} \]

**Case 1.5.** There was a request from at least one of the stops \( S_7 \) and \( S_8 \), as well as request from \( S_{13} \) (but no request from \( S_{14} \)). The sequence \( \text{RegularStops}^8 \) becomes \( \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_9, S_{10}, S_{11}, S_{12}, S_{13}\} \), the current route is updated as shown in Figure 3(c):

\[ S_2 - S_3 - S_4 - S_5 - S_6 - S_7 - S_9 - S_{10} - S_{11} - S_{12} - S_{13} \]

**Case 1.6.** There was a request from at least one of the stops \( S_7 \) and \( S_8 \), as well as request from \( S_{14} \) (in this case it does not matter whether there was a request from \( S_{13} \), because a request from \( S_{14} \) makes both \( S_{13} \) and \( S_{14} \) “regular” for this run). The sequence \( \text{RegularStops}^8 \) becomes \( \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}\} \), which also means that no further optional stop can be added to the sequence, because it already contains all the optional stops in the driving direction. The sequence remains unchanged until the end of the run. The current route is updated as shown in Figure 3(d):

\[ S_2 - S_3 - S_4 - S_5 - S_6 - S_7 - S_8 - S_{10} - S_{11} - S_{12} - S_{13} - S_{14} \]

**Case 2.** There was a request from the stop \( S_7 \) or \( S_8 \), but there were no requests from \( S_{13} \) and \( S_{14} \). The sequence \( \text{RegularStops}^8 \) for this case is \( \{S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_9, S_{10}, S_{11}, S_{12}\} \), and the route plan becomes
Sustainable Transport System

$S_1 - S_2 - S_3 - S_4 - S_5 - S_6 - S_7 - S_8 - S_9 - S_{10} - S_{11} - S_{12}$

In this case, stops $S_5$ and $S_9$ were already marked as “regular” for this run (the parameters $option_5$ and $option_9$ are set to be true), therefore we should not receive any further requests from these stops. Thus, when leaving $S_5$, we have three options for the updating of the current plan of the route.

Case 2.1. There were no requests from the stops $S_{13}$ and $S_{14}$. The sequence $RegularStopsR$ and the route plan remain unchanged.

Case 2.2. There was a request from the stop $S_{13}$, but there were no requests from $S_{14}$. The sequence $RegularStopsR$ becomes $[S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}]$, and the route plan becomes

$S_2 - S_3 - S_4 - S_5 - S_6 - S_7 - S_8 - S_9 - S_{10} - S_{11} - S_{12} - S_{13}$

Case 2.3. There was a request from the stop $S_{14}$. The sequence $RegularStopsR$ becomes $[S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}]$, the route plan becomes

$S_2 - S_3 - S_4 - S_5 - S_6 - S_7 - S_8 - S_9 - S_{10} - S_{11} - S_{12} - S_{13} - S_{14}$

Case 3. There was a request from the stop $S_{13}$, but there were no requests from $S_{12}$ or $S_{14}$. The sequence $RegularStopsR$ for this case is $[S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}]$, and the plan becomes

$S_1 - S_2 - S_3 - S_4 - S_5 - S_6 - S_7 - S_8 - S_9 - S_{10} - S_{11} - S_{12} - S_{13}$

In this case, stop $S_{13}$ is already marked as “regular” for this run (the parameters $option_{13}$ is set to be true), therefore we cannot receive any further requests from this stop. When leaving $S_5$, we can have four options for updating of the route plan.

Case 3.1. There were no requests from the stops $S_{12}, S_{13}$, and $S_{14}$. The sequence $RegularStopsR$ and the route plan remains unchanged.

Case 3.2. There was a request from $S_{12}$ or $S_{13}$, but no requests from $S_{14}$. The sequence $RegularStopsR$ becomes $[S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}]$, the plan becomes

$S_2 - S_3 - S_4 - S_5 - S_6 - S_7 - S_8 - S_9 - S_{10} - S_{11} - S_{12} - S_{13}$

Case 3.3. There was a request from $S_{14}$, but no requests from $S_{12}$ or $S_{13}$. The sequence $RegularStopsR$ becomes $[S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}]$, the plan becomes

$S_2 - S_3 - S_4 - S_5 - S_6 - S_7 - S_8 - S_9 - S_{10} - S_{11} - S_{12} - S_{13} - S_{14}$

Case 3.4. There was a request from $S_{12}$, as well as a request from $S_{14}$. The sequence $RegularStopsR$ becomes $[S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}]$, which also means that no further optional stop can be added to the sequence. The plan becomes

$S_2 - S_3 - S_4 - S_5 - S_6 - S_7 - S_8 - S_9 - S_{10} - S_{11} - S_{12} - S_{13} - S_{14}$
**Case 4.** There was a request from the stop $S_{14}$, but there were no requests from $S_7$ or $S_8$. The sequence $\textit{RegularStops}^k$ for this case is $[S_1, S_2, S_3, S_4, S_5, S_6, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}]$, and the plan becomes

$$S_1 - S_2 - S_3 - S_4 - S_5 - S_6 - S_9 - S_{10} - S_{11} - S_{12} - S_{13} - S_{14}$$

In this case, stops $S_{11}$ and $S_{12}$ were already marked as “regular” for this run (the parameters $\text{option}_{11}$ and $\text{option}_{12}$ are set to be true), therefore we cannot receive any further requests from these stops. Thus, when leaving $S_j$, we can have two options for the updating of the current plan of the route.

**Case 5.** There was a request from $S_7$ or $S_8$. The sequence $\textit{RegularStops}^k$ becomes $[S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}]$, which also means that no further optional stops can be added to the sequence. The plan becomes

$$S_1 - S_2 - S_3 - S_4 - S_5 - S_6 - S_7 - S_8 - S_9 - S_{10} - S_{11} - S_{12} - S_{13} - S_{14}$$

**Case 6.** There was a request from $S_7$ or $S_8$, as well as request $S_{14}$ (but not from $S_{13}$). The sequence $\textit{RegularStops}^k$ becomes $[S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}]$, and the route plan becomes

$$S_1 - S_2 - S_3 - S_4 - S_5 - S_6 - S_7 - S_8 - S_9 - S_{10} - S_{11} - S_{12} - S_{13}$$

In this case, stops $S_7, S_8$ and $S_{13}$ were already marked as “regular” for this run (the parameters $\text{option}_7, \text{option}_8$ and $\text{option}_{13}$ are set to be true), therefore we cannot receive any further requests from these stops. Thus, when leaving $S_j$, we can have two options for the updating of the current plan of the route.

**Case 5.1.** There were no requests from the stop $S_{14}$. The sequence $\textit{RegularStops}^k$ and the route plan remains unchanged.

**Case 5.2.** There was a request from the stop $S_{14}$. The sequence $\textit{RegularStops}^k$ becomes $[S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}]$, no further optional stop can be added to the sequence, and the route plan becomes

$$S_1 - S_2 - S_3 - S_4 - S_5 - S_6 - S_7 - S_8 - S_9 - S_{10} - S_{11} - S_{12} - S_{13} - S_{14}$$

**Case 6.** There was a request from $S_7$ or $S_8$, as well as from $S_{14}$. The sequence $\textit{RegularStops}^k$ becomes $[S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11}, S_{12}, S_{13}, S_{14}]$, no further optional stops can be added to the sequence, and the route plan becomes

$$S_1 - S_2 - S_3 - S_4 - S_5 - S_6 - S_7 - S_8 - S_9 - S_{10} - S_{11} - S_{12} - S_{13} - S_{14}$$

In this case, no further optional stops can be added to the sequence $\textit{RegularStops}^k$, the plan already contains all optional parts of the route (like in $\text{normalLoad}$ mode).

It is easy to see that even for a very small route with not very complicated optional parts there are many possibilities for options for the updating of the current plan of the route. Thus, it would be problematic for a driver to check for the timetable updates continuously and to analyse the current plan manually. It
Sustainable Transport System becomes especially complicated when a request to serve a particular stop implies that not only one, but many stops become “regular” for this run. This implies that a driver decision support system representing the updated plan of the route is required. A simplified driver view, as shown in Figure 2, improves the ease of system use by the driver and allows for focussing on the necessary current driving goals.

On-Demand Bus Scenario: Analysis of a Specific Bus Route

This example is based on the spatial information and the timetable of an existing real bus route 445, which operates in Melbourne area of Victoria, in Australia. Bus route 445 takes passengers between two railway stations at Hoppers Crossing and Werribee Station, cf. Figure 4. Depending on the particular time of the day, the whole route can take in the non-optimised case approximately 34-38 minutes in the direction of “Werribee Station to Hoppers Crossing” and 32-38 minutes in the direction of “Hoppers Crossing to Werribee Station” (according to the timetables on the webpage of the Public Transport Victoria (PTV), dated 27 July 2014).

Figure 4. Bus route 445 in Melbourne area, Australia (2014): Identification of the optional sections of the route and optimization of the route.
For the case when the bus is travelling in the direction to Hoppers Crossing Station, the bus departs from Werribee Station and runs via Manly St, Market Rd, Railway Av, Tarneit Rd, Thames Bvd, Hogans Rd, Tarneit Rd, Wilmington Av, Westmill Dr, Heaths Rd and Derrimut Rd to Werribee Plaza Shopping Centre; then via Derrimut Rd, Heaths Rd and Morris Rd to Hoppers Crossing Station.

For the case when the bus is travelling in the direction to Werribee Station, the bus departs from Hoppers Crossing Station and runs via Morris Rd, Heaths Rd and Derrimut Rd to Werribee Plaza Shopping Centre; then via Derrimut Rd, Heaths Rd, Westmill Dr, Wilmington Av, Tarneit Rd, Hogans Rd, Thames Bvd, Tarneit Rd, Railway Av, Market Rd and Manly St to Werribee Station, see Figure 4. In our representation, we use a short notation $S_0, \ldots, S_{30}$ for the stops.

All buses on this route provide wheelchair accessible services, which makes this route a good example for deviations along the route, for hours out of the peak times, to pick up disabled passengers with restricted capabilities.

We have identified optional sections of the route when the bus may be empty, by the dashed lines in Figure 4(a). To collect more precise and real time data about the route load, one can use approaches like (Fihn & Finndahl, 2011) which publishes information about the number of passengers at each stop along the tram route. For this scenario, we have not built the statistics by collecting real data, but by analysing the spatial information about the route 445 and other public transport routes in this area, as well as Google map information for this suburb. A larger case study using PTV statistics is planned for future work. People who live far from a railway station may choose to travel on public transport but not every day (housewives, school children or elderly people), or prefer to drive in their own private cars or taxis, if they need to go to another part of the city during the middle of the day or late evenings, i.e., times outside of peak hours. In general, the closer the bus stop is to the railway station, the more people catch the bus from this stop. Thus, the bus load on the route 445 is higher at the ends of the route but could be zero during some hours in the middle of the route, e.g., between stops $S_2$ and $S_{21}$ (which is the mainly residential part of the suburb) as well as between stops $S_{23}$ and $S_{25}$ (the stop $S_{24}$ is an appendix to the main road, to bring passengers to the Werribee Plaza Shopping Centre). This means, we do not need to optimize the ends of the route, but between stops $S_7$ and $S_{21}$ and between stops $S_{23}$ and $S_{25}$ an optimisation could be required. However, after analysis of the city map as well as the spatial information on other bus routes operating in this area, we recommend extending the ends of the route by additional optional stops at each end, $S_0$ (Werribee Station) and $S_{30}$ (Hoppers Crossing Station), namely $S_{w1}$, $S_{w2}$, $S_{w3}$ and $S_{H1}$. Thus, the stops $S_0$ and $S_{30}$ become the provisional ends of the route, where the new actual ends of the route are $S_{w3}$ (Werribee Sport Centre) and $S_{H1}$ (Victoria University).

We present the extended and optimized route in Figure 4(b). The dotted lines represent possible shortcuts of the route.

In the rest of this section we discuss particular scenarios that are possible for this route in the irregularLoad mode. For simplicity, we are focussing only on the direction $S_{w3} - S_{H1}$, i.e. from the Werribee Sport Centre to the Victoria University. Here we have four parts of the route that could be optimized:

1. $S_{w3}$ to $S_6$
2. $S_5$ to $S_{21}$
3. $S_{23}$ to $S_{25}$
4. $S_{30}$ to $S_{H1}$
For this scenario, we can deal with the parts at the ends of the route in the same way as for the general scenario. The case of the stop \( S_{24} \), which is an appendix part of the route, is trivial. Thus, we focus here on the most interesting optional part of the route, between stops \( S_5 \) and \( S_{21} \).

When the bus is approaching the stop \( S_5 \), we could have seven possible cases for the bus driving scenarios, cf. also Figures 5 – 7 where we present the parts of the route between stops \( S_5 \) and \( S_{21} \) for these scenarios. Please note that we present here scenarios for system views – as we do not present system views for a concrete moment of time, we omit here the information about departure times and actual/planned location of the bus. However, the simplified driver views are presented assuming a concrete moment of time during the route run.

**Figure 5. Bus route 445: Possible scenarios after the stop \( S_5 \) (Cases 1, 2 and 3.1)**
**Case 1.** There are no requests for the whole part of the route. In this case, the bus should go directly to the stop $S_{21}$, following Tarneit Rd until the intersection with Heaths Rd, then turning on to Heaths Rd. For the graphical representation cf. the route in Figure 5, Case 1.

**Case 2.** There are requests only from the stop $S_p$ and/or $S_q$. The bus proceeds to the stop $S_q$ (via $S_p$), and then goes directly to the stop $S_{21}$ via Heaths Rd (cf. the route in Figure 5, Case 2).

**Case 3.** There are requests from the stop $S_p$ and/or $S_q$. The bus proceeds directly to the stop $S_q$ (via $S_p$, $S_q$, and $S_p$), and the rest of the route plan depends on the following five options (for all these options, the simplified driver view is the same, cf. the simplified driver view for the Case 3.1 in Figure 5):

**Case 3.1.** There are no further requests from the stops $S_{10}, S_{11}, ... , S_{20}$. The bus proceeds directly to the stop $S_{21}$, following Prospect Drive until the intersection with Tarneit Rd, then via Tarneit Rd until the intersection with Heaths Rd, then turning on to Heaths Rd, (cf. the route in Figure 5).

**Case 3.2.** There are some requests from the stops $S_{10}, S_{11}$, and/or $S_{12}$, but no requests from $S_{13}, S_{14}$, and $S_{15}$. From $S_p$, the bus proceeds to the stop $S_{12}$ (via $S_{10}$ and $S_{11}$), and then we have three additional options:

1. There were no requests from $S_{16}, \ldots , S_{20}$. From $S_{12}$, the bus proceeds to $S_{21}$ following the Hope Way until the intersection with Tarneit Rd, then via Tarneit Rd until the intersection with Heaths Rd, then turning on to Heaths Rd as shown in Figure 6, Case 3.2.1.
2. There is a request from $S_{16}$. From $S_{12}$, the bus proceeds to $S_{16}$ following the Hope Way. After $S_{16}$ it proceeds to $S_{21}$ like in normalLoad mode, as shown in Figure 6, Case 3.2.2.
3. There is no request from $S_{16}$, but there is a request at least from one of the stops $S_{17}, \ldots , S_{20}$. From $S_{12}$, the bus proceeds to $S_{17}$ following the Hope Way, until the intersection with Tarneit Rd, then via Tarneit Rd until the intersection with Wilmington Av, and then turning on to Wilmington Av. After $S_{17}$ it proceeds to $S_{21}$ as in normalLoad mode, as shown in Figure 6, Case 3.2.3.

**Case 3.3.** There are some requests from the stops $S_{13}, S_{14}$, and/or $S_{15}$ (for this case, it is not important whether there are requests from $S_{10}, S_{11}$, and/or $S_{12}$). From $S_p$, the bus proceeds to $S_{12}$ (via $S_{10}, \ldots , S_{15}$ as in normalLoad mode). The rest of the route plan until $S_{21}$ depends on three following options:

1. There is no request from $S_{16}, \ldots , S_{20}$. From $S_{12}$, the bus proceeds to $S_{21}$ following Tarneit Rd, then turning to Heaths Rd Av, cf. Figure 6, Case 3.3.1.
2. There is a request from $S_{16}$. From $S_{12}$, the bus proceeds to $S_{21}$ (via $S_{17}, \ldots , S_{20}$) as in normalLoad mode, as shown in Figure 7, Case 3.3.2.
3. There is no request from $S_{16}$, but there are requests from $S_{17}, \ldots , S_{20}$. From $S_{12}$, the bus proceeds to $S_{17}$ following Tarneit Rd until the intersection with Wilmington Av, then turning on to Wilmington Av. After $S_{17}$ it proceeds to $S_{21}$ as in normalLoad mode, cf. Figure 7, Case 3.3.3.

**Case 3.4.** There are no requests from the stops $S_{10}, S_{11}, \ldots , S_{15}$, but there is a request from $S_{16}$. From $S_p$, the bus proceeds directly to the stop $S_{16}$, following Prospect Drive, then turning on to Tarneit Rd, then via Tarneit Rd until the intersection with Hope Way, then turning on to Hope Way. After $S_{16}$ it proceeds to $S_{21}$ as in normalLoad mode. The system view on the route is shown in Figure 7, Case 3.4.

**Case 3.5.** There are no requests from the stops $S_{10}, S_{11}, \ldots , S_{16}$, but there is a request from at least one of the stops $S_{17}, \ldots , S_{20}$. From $S_p$, the bus proceeds directly to the stop $S_{17}$, following Prospect Drive, then
Figure 6. Bus route 445: Possible scenarios after the stop $S_5$ (Cases 3.2.1 – 3.3.1, system views)
Figure 7. Bus route 445: Possible scenarios after the stop $S_5$ (Cases 3.3.2 – 3.5, system views)
Sustainable Transport System

turning on to Tarneit Rd, then via Tarneit Rd until the intersection with Wilmington Av, then turning on to Wilmington Av. After $S_j$, it proceeds to $S_{j2}$ as in *normalLoad* mode. The system view on the route is shown in Figure 7, Case 3.5.

**Case 4.** There are no requests from the stop $S_{i\gamma'}$ and/or $S_{i\gamma''}$ but there is a request from at least one of the stops $S_{i10}, S_{i11}, S_{i12}$. The bus goes to the stop $S_{i12}$ (via $S_{i\gamma'}, \ldots, S_{i11}$), and after that we have four additional options:

- **Case 4.1.** There are no requests from $S_{i13}, \ldots, S_{i20}$. From $S_{i12}$, the bus proceeds to the stop $S_{i21}$ following Hope Way until the intersection with Tarneit Rd, then via Tarneit Rd until the intersection with Heaths Rd, then turning on to Heaths Rd. This scenario is equal to the Case 3.2.1, cf. Figure 6.

- **Case 4.2.** There is a request from at least one of the stops $S_{i13}, \ldots, S_{i15}$. From $S_{i12}$, the bus proceeds to the stop $S_{i15}$ following Hope Way. After $S_{i15}$, it proceeds to $S_{i21}$ as in *normalLoad* mode. This scenario is equal to the Case 3.2.2, cf. the corresponding system view in Figure 6.

- **Case 4.3.** There are no requests from $S_{i13}, \ldots, S_{i15}$, but there is a request from $S_{i16}$. From $S_{i12}$, the bus proceeds to the stop $S_{i16}$ following Hope Way until the intersection with Tarneit Rd, then via Tarneit Rd until the intersection with Heaths Rd, then turning on to Heaths Rd. This scenario is equal to the Case 3.2.3, cf. Figures 6 and 7 for the corresponding system views.

- **Case 4.4.** There are no requests from $S_{i13}, \ldots, S_{i15}$, but there is a request from at least one of the stops $S_{i16}, \ldots, S_{i20}$. From $S_{i12}$, the bus proceeds to $S_{i16}$ following Hope Way until the intersection with Tarneit Rd, then via Tarneit Rd until the intersection with Wilmington Av, then turning on to Wilmington Av. After $S_{i16}$, it proceeds to $S_{i21}$ as in *normalLoad* mode. This scenario is equal to the Case 3.2.4 (cf. Figure 7 for the system view).

**Case 5.** There are no requests from the stops $S_{i\gamma'}, \ldots, S_{i22}$, but there is a request from at least one of the stops $S_{i3}, \ldots, S_{i15}$. From $S_{i}$, the bus proceeds to the stop $S_{i15}$ as in *normalLoad* mode. The scenarios for the rest of the route until $S_{i21}$ are as described for the Cases 3.3.1 – 3.3.3, cf. Figures 6 and 7 for the corresponding system views.

**Case 6.** There are no requests from the stops $S_{i\gamma'}, \ldots, S_{i15}$, but there is a request at least from one of the stops $S_{i16}, \ldots, S_{i17}$. From $S_{i}$, the bus proceeds directly to the stop $S_{i16}$ following Tarneit Rd until the intersection with Wilmington Av, then turning on to Wilmington Av. After $S_{i16}$, it proceeds to $S_{i21}$ as in *normalLoad* mode. This route in this case is equal to the route described for the Case 3.4 (cf. Figure 7 for the system view).

**Case 7.** There are no requests from the stops $S_{i\gamma'}, \ldots, S_{i16}$, but there is a request from at least one of the stops $S_{i17}, \ldots, S_{i20}$. From $S_{i}$, the bus proceeds directly to the stop $S_{i17}$ following Tarneit Rd until the intersection with Wilmington Av, then turning on to Wilmington Av. After $S_{i17}$, it proceeds to $S_{i21}$ as in *normalLoad* mode. This route in this case is equal to the route described for the Case 3.5 (cf. Figure 7 for the system view).

We are interested in the required *on-demand request time* for this route. We assume that the bus should take 3 minutes to go from stop $S_{30}$ to stop $S_{H1}$ or from $S_{H1}$ to $S_{30}$, as well as 5 minutes from stop $S_{0}$ to stop $S_{W3}$ or from $S_{W3}$ to $S_{\gamma'}$. According to the current timetables, the bus needs 11 minutes to go from $S_{i}$ to $S_{i21}$ in *normalLoad* mode. Thus, the required request time for this route would be 11 minutes. We round this time up to the nearest 5 minutes and obtain the *on-demand request time* for this route: 15 minutes. This means that during the *irregularLoad* mode the request for the bus to come to a stop on an optional part of the route should come at least 15 minutes earlier than the time specified in the bus timetable.

As an example of this irregular route, let us consider the stop $S_{14}$ (Harmony Dr/Hogans Rd) and analyse the timetable for Monday to Friday, assuming the *irregularLoad* mode is between 10am and 2:30pm, as well as after 8pm (marked bold). We find the following for the given times:
Thus, a passenger, wishing to catch the bus from this stop, at 11:01am, should send a request to the on-demand bus system no later than at 10:46am. To make it easier for passengers, we could add the on-demand request times to the timetable in the following manner:

6:22, 7:16, 8:11, 9:01,

10:01, 11:01, 12:01, 1:01, 2:01,

3:01, 4:01, 5:01, 5:58, 7:04,

8:08, 9:08.

These ideas could be implemented as a mobile application that allows passengers to send their on-demand requests to the driver, as well as a decision support application for the driver.

By serving the optional parts of this route only on-demand, we can optimise the timetables making them denser, i.e., enabling the bus to pass by, not every 60 minutes approximately, but every 30 minutes instead.

**Implementation**

To make this model really useful for the passengers and for the drivers, the system must be implemented as a smartphone or web application. The first prototype of the system was implemented within the “Buses on demand” student project at our School of Computer Science and IT in RMIT University, Australia.

The prototype allows passengers to send requests to the transport system any time and from any location, choosing for the *irregularLoad* mode which “optional” stops should be operated, as well as see which stops have already been “requested” by other passengers, i.e., which stops are currently marked to be “regular” (cf. Figure 8 for the passenger view of the system and for the graphical representation of the system view). To increase the safety on the road, an automated decision support system was implemented: the prototype also provides the driver with instructions, on how the route should be shortcut if optimisation is required. Thus, the prototype provides the driver with the latest version of the timetable and the optimized route: the driver’s view of the route comprises of only the “regular” stops, and when an “optional” stop becomes “regular”, the driver’s view of the route is immediately updated. The system view provides additional information to the system manager on the requests statistics. The prototype supports two kinds of representations: graphical, using Google maps, as presented in Figure 8, and textual, using tabular representation.
Figure 8. Prototype of the bus on-demand system: passenger view (the “print screen” image is cut in the middle) and the graphical representation of the route using Google maps (red markers denote “regular” stops, blue markers denote “optional” stops).
FUTURE RESEARCH DIRECTIONS

Currently the buses in the public transport system in Melbourne do not have GPS devices, or if the buses are new and GPS has been fitted, it is turned off. There is the possibility that the positions of buses fitted with GPS tracking technology can be integrated into the wider public transport system, cf. (Covan, 2014). This would be a tremendous advantage for our proposed on-demand model, and would allow for integrating the modelling of exceptions on the road (such as problems with traffic due to road works, car accidents, etc.) into the model directly. In further research on the modelling of on-demand systems, we plan a number of case studies based on analysis of actual train and tram routes in Melbourne, Australia.

There are further opportunities to develop mobile applications to notify commuters if their trains or buses are late. Such systems might rely on the vehicle with the GPS tracking device to arrive at a particular stop and notify people using the mobile application that it has arrived. This reminder could be used to predict when the bus would arrive at their stop and remind them within 15 minutes of reaching it. The time could be flexible and set by the passenger to enable them to reach their stop in whatever time is required to catch the bus.

The GPS could also be used to remind people travelling in the bus about when they are nearing their destination. For disabled people, or commuters engrossed in their reading material or games, they may otherwise have missed the stop, or not managed to get to the exit door in time.

There are opportunities for more complex multimodal journey planning, which would notify passengers that they are running late and will miss their connecting train. Such a system would inform the passengers of other transport options, which may take them to their destinations using different modes of transport, if they were to get off a stop earlier perhaps and connect in a different place. There are other options for personalisation, for instance, for wheelchair passengers, to let them know which buses and stops enable easy access for wheelchairs. There are safety options for encouraging commuters to notify selected people of their whereabouts, when they are approaching a particular stop to meet them, or when they are setting out on their journey to inform them of the time of their expected arrival. There are sustainable options for encouraging passengers to walk parts of their journey for exercise and to encourage fitness, or to by-pass congested areas and consider other options. These options can be also combined with our recent work on autonomous systems, cf. (Spichkova & Simic, 2015).

It may be that people would like to be informed of other transport options, which exist near their current stop, or position. This may happen in the case of a traffic jam or accident, where the bus is caught in a particular street and cannot continue its journey for some time. There may be nearby optional routes on other forms of transport, which could avoid this congestion.

CONCLUSION

The overall contribution of this chapter has been the development of a formal model for an on-demand transport system. The system we have developed here can be seen as a part of a smart city, and is much more flexible and sustainable in comparison to the existing traditional one, enabling the reduction of greenhouse gas emission and noise pollution, as well as the saving of fuel costs for the company. Moreover, the presented solution allows having a more flexible/dense timetable and also possible longer routes as well as subjective shorter travelling times.
Sustainable Transport System

The key features of the defined on-demand transport system include the optionality of particular stops along the route and at either end of the route, with the specification based on the actual load modes of the transport system. However, the development of the on-demand transport system is more complicated, requiring a large number of additional aspects to be taken into account, and a DSS to be built to enable the driver to react promptly and effectively to the dynamic changes in the timetables. In this case, a proper formal model is required to build the system correctly.

Our model covers several aspects of sustainable transport and focuses on spatial planning and optimization including environmental issues in transport planning. The model incorporates three distinct views of operation:

1. The driver view: This provides a list of all the stops, optional and regular, where the bus is expected to travel at the current moment in time. The route plan is updated dynamically, but there is a DSS embedded in this view, so the driver will not be distracted while driving, but keep a regular update on whether some parts of the route need to be skipped or changed during that particular run. From the DSS, the driver always has the latest version of the timetable for his route and the information that is currently required to complete the run in the most efficient and safest way. To increase the usability, we have introduced two modes of this view: the general mode where the driver can overview the complete current route plan, where the simplified mode contains the information about 4 stops/stations only: the current stop, where the vehicle is/should be approaching, and three next consecutive stops. This allows the driver to focus on the current part of the route, where the DSS takes control over the further updates of the plan.

2. The passenger view: This describes an interface for passengers to provide real-time information about the transport system. It enables them to select a particular route and to request the system to book the bus to stop at any optional parts of the route.

3. The system view: this provides a map of the overall route where the bus is travelling. It has identified the regular stops and includes any optional stops, which have been selected by passengers for the requested time slot.

The model, various scenarios and optimizations have been evaluated as case studies based on the analysis of an actual bus route 445 in Melbourne area, Australia (Public Transport Victoria, 2014). The current integrated system described above exists as a prototype only, and requires incorporation and testing on a real time bus system. If fully implemented, across all modes of transport, this system could provide for a much more sustainable, flexible and user-friendly public transport system. If passengers knew such a system existed, they might be encouraged to travel, especially the optional routes, and this might stimulate potential passengers to become actual ones and to switch from private transport to public.

REFERENCES


KEY TERMS AND DEFINITIONS

(Public Transport) Vehicle: A bus, a tram, a train, a ferry, or a car used as a public transport proposes. In this chapter we restrict this definition to buses, trams and trains.

Irregular Connection: A part of the public transport route, which can be empty over some period of time during the hours of public transport operation.

Irregular Load Mode of a Route: A mode of operation when the vehicle could be empty, i.e. have idle drive, for part of the route.

Mode: A set of behavioural properties, which is valid in a particular context.

Normal Load Mode of a Route: A mode of operation when the vehicle is not empty for any part of the route.

On-Demand Request Time: a predefined time, which is specified by the public transport system as a deadline for sending requests (during the irregular load mode of the route operation) for attending the vehicle on an optional part of the route.
Sustainable Transport System

Optional Part of the Route: a part of the public transport route (more precisely, of its irregular connections, cf. above), which can be shortcut between two provisional shortcut stops (if it is in the middle of the route) or skipped from the provisional to the actual end of the route (if it’s at the end of the route).

Provisional End of the Route: A stop/station, where the regular connection ends for this direction of the route.

Provisional Shortcut Stop of the Route: A stop/station, from which the public transport vehicle can shortcut an irregular stop when the public transport route is in irregular load mode.