U.PORTO

PROGRAMA DOUTORAL EM ENGENHARIA INFORMÁTICA

ON IMPROVING OPERATIONAL PLANNING AND CONTROL IN PUBLIC TRANSPORTATION NETWORKS USING STREAMING DATA: A MACHINE LEARNING APPROACH

Luís Alexandre Moreira Matias

D

2014

Luís Alexandre Moreira-Matias

On Improving Operational Planning and Control in Public Transportation Networks using Streaming Data: A Machine Learning Approach

Tese apresentada à Faculdade de Engenharia da Universidade do Porto
para obtenção do grau de Doutor em Ciências de Engenharia,
realizada sob orientação científica do
Prof. Doutor João Pedro Carvalho Leal Mendes Moreira,
Professor Auxiliar da Faculdade de Engenharia da Universidade do Porto,
e co-orientação científica do
Prof. Doutor João Manuel Portela da Gama,
Professor Associado da Faculdade de Economia da Universidade do Porto

Departamento de Engenharia Informática
Faculdade de Engenharia da Universidade do Porto

Janeiro de 2015

Dedico esta tese aos meus pais e avós.

Resumo

O objetivo desta tese é melhorar o Planeamento e o Controlo das Redes de Transportes Públicos Rodoviários (i.e. autocarros e táxis) utilizando os dados geoespaciais transmitidos por cada um dos veículos em tempo real. Para tal, propomo-nos a monitorizar as operações dessas redes no sentido de obter informação que ajude, de alguma forma, a prever os seus futuros estados a curto/médio prazo. Numa primeira abordagem, foi feito um estudo do Estado da Arte em métodos baseados em dados geoespaciais para resolver problemas relacionados com este tópico. Com este passo, pretendemos identificar problemas de investigação em que este tipo de métodos possa, de alguma forma, sofrer e/ou proporcionar algum avanço.

Para resolver tais problemas, propomo-nos a desenvolver algoritmos e/ou metodologias sustentáveis (dum ponto de vista computacional) para lidar com estas fontes de grandes quantidades de dados. Estes algoritmos irão contribuir para uma melhoria na qualidade dos serviços de transportes públicos o que irá, em última análise, melhorar a mobilidade do Homem nas grandes áreas urbanas.

Na sequência do referido estudo do estado da arte, foi possível identificar três problemas concretos onde este tipo de dados representa uma mais valia: (1) Avaliação Automática da Cobertura de Horários em Transportes Coletivos; (2) Redução em Tempo Real das ocorrências de *Bus Bunching* (i.e. agrupamentos indesejados de veículos de Transportes Coletivos) e (3) Recomendações Inteligentes em Tempo real sobre a praça de Táxis mais conveniente para ir em cada momento (do ponto de vista dos condutores), de acordo com o presente estado da rede. No sentido de resolver cada um destes problemas, propuseram-se diferentes metodologias de aprendizagem automática que superam e/ou complementam, de alguma forma, as soluções que já existem atualmente.

O primeiro problema (1) é relativo aos dias que são cobertos pelo mesmo plano de horário. Normalmente, Esta definição feita durante o planeamento das rotas e é baseada na relação entre os perfis de procura gerados nesta etapa e os recursos disponíveis para servir essa mesma procura. Em consequência deste facto e tanto quanto sabemos, não existe qualquer trabalho de investigação neste tópico. Todos os dias cobertos pela mesmo plano de horário possuem exactamente o mesmo perfil diário (i.e. a duração total das viagens em função da sua hora de partida) pois eles partilham esse mesmo horário. Todavia, os valores reais desses tempos de viagem pode divergir dos agendados (provocando, desta forma, uma discrepância indesejada entre os tempos de viagem no horário e aqueles vividos pelos passageiros no dia-a-dia). Para ultrapassar este problema, propôs-se avaliar se a cobertura de cada horário é adequada ao comportamento

dos veículos no dia-a-dia. Esta avaliação foi feita utilizando uma nova metodologia de aprendizagem automática desenvolvida especialmente para este efeito. Esta ferramenta explora as diferenças entre os horários teóricos e os reais para juntar cada um dos dias em diferentes grupos. Este agrupamento automático e feito de acordo com uma métrica de distancia calculada sobre os seus perfis. Em seguida, uma ferramenta de indução e utilizada para extrair regras compreensíveis sobre que dias devem ser cobertos por cada plano de horário. Tais regras podem ser utilizadas pelos responsáveis de planeamento para proporem alterações a referida cobertura.

A ocorrência de (2) *Bus Bunching* (BB) é um dos indicadores mais evidentes da falta de fiabilidade dum serviço de transporte públicos. Dois (ou mais) autocarros a circular juntos na mesma rota e um sinal inegável de que algo corre bastante mal nesse mesmo serviço. Tipicamente, o estado da arte neste tópico passa por assumir que a probabilidade da ocorrência de BB e minimizada por garantir a estabilidade da frequência entre veículos. Apesar de valida, esta abordagem implica adotar múltiplas ações corretivas (ex.: redução da velocidade máxima ou aumento do tempo de paragem). Consequentemente, estes processos impõe uma sobrecarga de trabalho mental para os condutores que podem, em ultimo caso, não ser capazes de cumprir tais ordens. Nesse sentido, propusemos adoptar uma abordagem pro-activa a este problema de controlo operacional – em oposição as referidas táticas reativas. A ideia passa por estimar a probabilidade da ocorrência de BB nas futuras paragens da rota que, caso ultrapasse um determinado limiar, lança um alarme que pode resultar na recomendação duma acção de controlo para evitar tal ocorrência. Esta probabilidade é calculada através dum novo método incremental de aprendizagem automática supervisionada desenvolvido especificamente para este efeito. Este método explora simultaneamente a base de dados geoespaciais da rota e os dados transmitidos por cada veículo em tempo real para, não só reduzir as ocorrências de BB, mas também a quantidade de recursos necessários para efectuar estas decisões. Este método inspira-se em diversos metodologias estatísticas e de optimização tais como redes neuronais, métodos de regressão linear/não linear e teoria de probabilidades, entre outros.

A (3) inteligência de mobilidade dos condutores de táxis é um fator fundamental para maximizar tanto a fiabilidade do serviço, bem como a sua rentabilidade. Naturalmente, o conhecimento sobre onde e quando a procura de serviços de táxi vai emergir acrescenta uma larga vantagem competitiva ao condutor – especialmente em cenários onde não é viável, dum ponto de vista económico, conduzir o veículo aleatoriamente pela cidade até encontrar o próximo passageiro. A escolha da próxima praça onde parquear o veículo para aguardar pelo próximo serviço baseia-se em quatro variáveis: (i) a o preço esperado dum serviço nas praças ao longo do tempo; (ii) a distância, em tempo ou em espaço, entre a sua posição atual e a posição das praças; (iii) o numero de táxis já posicionados nas praças neste momento e (iv) a procura em cada praça ao longo do tempo. Contudo, tanto quanto sabemos, não existe qualquer trabalho de investigação que considere estas quatro variáveis simultaneamente. O valor da variável (iii) pode ser obtido diretamente com base na posição dos veículos da frota em tempo real – todavia, as restantes três variáveis requerem modelos de previsão a curto prazo para podermos estimar o seu valor.

A tipificação da procura de serviços de táxi a curto prazo é um problema complexo. Essa procura pode ser decomposta em (iv) quantidade de passageiros (i.e. um número inteiro positivo) e (i) o valor esperado do preço desses mesmos serviços (i.e. uma categoria de preços). No sentido de resolver este problema, propôs-se uma metodologia que utiliza alguns métodos de análise de series temporais em conjunto com técnicas de discretização. Esta metodologia distingue-se das já existentes por ser capaz de aprender incrementalmente e, desta forma, adaptar-se adequadamente a qualquer cenário de procura (ex.: picos de procura inesperados).

A variável (ii) diz respeito à quantidade de tempo necessária para chegar a um ponto da cidade/praça onde existem condições de procura favoráveis (ex.: uma elevada procura ou uma procura de elevada rentabilidade). Consequentemente, este problema consiste numa previsão do tempo de viagem apriori. A previsão do tempo de viagem é um problema muito estudado na literatura. Tipicamente, este problema é resolvido com a aplicação direta um algoritmo de aprendizagem supervisionada já existente. Aqui, decidiu-se apresentar uma abordagem mais genérica e complexa a este conhecido problema. Esta decisão baseia-se em dois fatores: (ii-1) a necessidade criar uma forma sustentável de extrair tanta informação quanto possível dos dados transmitidos por estas redes veiculares, independentemente do tipo conhecimento (i.e. variável objetivo) que queremos extrair; (ii-2) garantir que somos capazes incluir múltiplas fontes de dados para conseguir melhorar a quantidade de informação disponível para os modelos de aprendizagem. Para resolver este problema neste contexto, propôs-se uma nova metodologia para manter estatísticas suficientes sobre uma ou varias variáveis de interesse sobre uma matriz de origem-destino dinâmica ao longo do tempo. Esta metodologia inclui técnicas de aglomeração de dados geoespaciais e algoritmos de aprendizagem automática incrementais.

Todos estes problemas foram resolvidos utilizando dados reais transmitidos pelos dois maiores operadores de transportes públicos rodoviários a operar no Porto (Portugal). Estas *frameworks* atingiram resultados promissores nas experiencias que foram efetuadas com recurso a esses mesmos dados, demonstrando assim a sua utilidade no mundo real. Este trabalho resultou em dezasseis publicações de elevada qualidade em conferências e revistas reconhecidas internacionalmente.

Abstract

This thesis is focused on improving both Operational Planning and Control of Public Road Transportation (PT) Networks (i.e. buses and taxis) using location-based data gathered through the Global Positioning System (GPS data). Its aim is to monitor the operations of these vehicular networks to infer useful information about their future status on both short-term and long-term horizons. To do it so, we undertook an explorative approach by surveying the data driven methods on this topic in order to identify research opportunities worthy to be further studied. The main idea is to provide sustainable frameworks (in a computational point of view) to handle this massive sources of data. Ultimately, we want to extract information useful to improve Human Mobility on the major urban areas.

As result of the abovementioned survey, three concrete problems were addressed on this thesis: (1) Automatic Evaluation of the Schedule Plan's Coverage; (2) Real-Time Mitigation of Bus Bunching occurrences; (3) Real-Time Smart Recommendations about the most adequate stand to head to in each moment according to the current network status. To do it so, we developed Machine Learning (ML) frameworks in order to advance the State-of-The-Art on such problems.

The first problem (1) concerns the days that are covered by the same schedule. This definition is usually made during the design of the network planning and it is based on the relationship between the demand profiles generated and the resources available to meet such demand. Consequently, at the best of our knowledge, there is no research work addressing this topic using GPS data. All the days covered by the same timetable have exactly the same daily profile due to the fact that they share the same departing/arrival times. However, the real values of such times may differ from the original ones (causing an undesired gap between the defined timetables and the real ones). To overcome this issue, we propose to evaluate if such coverage still meets the network behavior using a ML framework. It explores such differences by grouping each one of the days available into one of the possible coverage sets. This grouping is made according to a distance measured between each pair of days where the criteria rely on their profiles. As output, rules about which days should be covered by the same timetables are provided. Such rules can be used by the operational transportation planners to perform the abovementioned evaluation. These rules also provide insights on how the current coverage can be changed in order to achieve that.

The prevalence of (2) Bus Bunching (BB) is one of the most visible charac-

teristics of an unreliable service. Two (or more) buses running together on the same route is an undeniable sign that something is going terribly wrong with the company's service. Most of the state-of-the-art on this topic departs from the assumption that the probability of BB events is minimized by maximizing headway stability. Notwithstanding its validity, this approach requires multiple control actions (e.g. speed modification, bus holding, etc.) which may impose high mental workload for drivers and result with low compliance rates. Hereby, we propose a proactive rather than a reactive operational control framework. The basic idea is to estimate the likelihood of a BB event occurring further downstream to then let an event detection threshold triggers the deployment of a corrective control strategy. To do it so, we propose a Supervised Online Learning framework. It is focused on exploring both historical and real-time AVL data to build automatic control strategies, which can mitigate BB from occurring while reducing the human workload required to make these decisions. State-of-the-art tools and methodologies such as Regression Analysis, Probabilistic Reasoning and Perceptron constitute building blocks of this predictive methodology.

The (3) taxi driver mobility intelligence is an important factor to maximize both profit and reliability within every possible scenario. Knowledge on where the services (transporting a passenger from a pick-up to a drop-off location) will actually emerge can be an advantage for the driver - especially when there is no economic viability of adopting random cruising strategies to find passengers. The stand-choice problem is based on four key variables: (i) the expected revenue for a service over time, (ii) the distance/cost relation with each stand, (iii) the number of taxis already waiting at each stand and (iv) the passenger demand for each stand over time. However, at the best of our knowledge, there is no work handling this recommendation problem by using these four variables simultaneously. The variable (iii) can be directly computed by the real-time vehicle's position - however, the remaining three need to be estimated for a short-term time horizon.

To estimate the short-term demand that will emerge at a given taxi stand is a complex problem. Such demand can be decomposed into two axis: the (iv) pick-up quantity (i.e. an integer representing the number of services to be demanded) and (i) the expected revenue for a service over time (i.e. a fare-based category). To do it so, we propose a framework based on both time series analysis and discretization techniques which are able to perform such supervised learning task incrementally.

The variable (ii) is related on how much time it will take to get to a given urban area/taxi stand where there are favorable service demand conditions (e.g. high service demand in terms of passenger quantity or revenue-based). Consequently, it is focused on apriori Travel Time Estimation. This problem is vastly covered on the literature - namely, by using Regression analysis. However, we propose a most general technique to address this problem. There are two motivations to do it so: (ii-1) to provide a sustainable way to handle these large amount of data in order to extract usable information from it independently of the problem we want to solve (namely, its variable of interest); (ii-2) to be able to include multiple data sources in order improve the penetration rate (i.e. the ratio of ground truth information) of our framework. To carry out such task, we propose incremental discretization techniques to maintain accurate statistics of interest over a time-evolving Origin-Destination matrix. These techniques include spatial clustering and incremental ML algorithms.

All these problems were addressed using real world data collected from two major public road transportation companies running in Porto, Portugal. These frameworks achieved promising results on the experiments conducted to validate them. This work resulted into sixteen high quality peer-reviewed publications at internationally known venues and journals.

Résumé

L'objectif de cette thèse est d'améliorer la planification et le contrôle des réseaux de transport public routier (tels que les bus et les taxis) en utilisant des données géo-spatiales recueillies par les appareils GPS (*Global Positioning System*).

D'abord, il faut surveiller les activités de ces réseaux afin d'obtenir des informations que puissent aider à prévoir leurs futurs états dans le court / moyen terme. Dans une première approche, on a dû conduire une étude de l'état de l'art à propos des méthodes basées sur les données géo-spatiales pour résoudre les problèmes liés à ce sujet. Le but de cette étape est d'identifier les problèmes de recherche où telles méthodes pourraient en quelque sorte être utiles et fournir une certaine amélioration. L'objectif principal de ce travail est de développer des algorithmes et / ou des méthodes durables (du point de vue informatique) pour faire face à ces sources de grandes quantités de données.

En fin de compte, l'objectif est de faire progresser la mobilité de l'homme dans les grandes zones urbaines. L'analyse de l'état de l'art a rendu possible d'identifier trois problèmes spécifiques où ces données représentent une grande valeur ajoutée: (1) l'évaluation automatique de la couverture des horaires dans les transports en commun; (2) la diminution en temps réel des occurrences de groupage de bus (groupements indésirables des véhicules de transport en commun) et (3) les recommandations intelligentes en temps réel sur la station de taxi la plus pratique pour aller à tout moment (du point de vue des chauffeurs) selon l'état actuel du réseau. Pour résoudre ces problèmes, on propose différentes méthodes d'apprentissage automatique qui dépassent, en aucune manière, les solutions qui existent déjà aujourd'hui.

Le premier problème (1) est à propos des jours qui sont couverts par le même plan horaire. Normalement, ce réglage est effectué lors de la planification des itinéraires et est basé sur le ratio entre les profils de la demande générée dans cette étape et les ressources disponibles pour servir cette demande. En conséquence, et aussi loin que nous le savons, il n'y a aucun travail de recherche sur ce sujet. Chaque jour couvert par le même plan horaire a exactement le même modèle quotidien (c'est-à-dire, la longueur totale des voyages en fonction de leur heure de départ) parce qu'ils partagent le même horaire. Pourtant, les valeurs réelles de ces temps de voyage peuvent différer de ceux prévus (provoquant ainsi un décalage indésirable entre les temps de voyage du plan et ceux subis chaque jour par les passagers).

La survenue de (2) groupage de bus (BB) est l'un des indicateurs les plus clairs de l'absence de fiabilité d'un service de transport public. Deux (ou plus) bus fonctionnant ensemble sur le même itinéraire est un signe indéniable que

quelque chose ne va pas bien dans le service. Typiquement, l'état de la technique dans cette discussion suppose que la probabilité d'occurrence de BB est réduite au minimum en assurant la stabilité de fréquence des véhicules. Bien que valide, cette approche implique l'adoption de plusieurs mesures correctives (ex: réduction de la vitesse maximale ou accroissement du temps d'arrêt). Par conséquence, ces procédés nécessitent d'une surcharge de travail mentale pour les pilotes qui peuvent, à terme, n'être pas en état d'accomplir ces commandes. Dans ce sens, nous nous proposons d'adopter une approche proactive à ce problème de contrôle opérationnel - par opposition à celles tactiques réactives. L'idée est d'estimer la probabilité de BB dans les arrêts futurs de la route et, si elle dépasse un certain seuil, déclencher une alarme qui peut aboutir à la recommandation d'une action de contrôle pour prévenir un tel événement. Cette probabilité est calculée en utilisant une nouvelle méthode d'apprentissage automatique supervisée incrémentale développée spécifiquement à cette fin. Cette méthode explore simultanément la base de données géo-spatiales de la route et les données transmises par chaque véhicule en temps réel non seulement pour réduire les occurrences BB mais aussi la quantité de ressources nécessaires pour prendre ces décisions. Cette méthode s'inspire sur diverses méthodologies statistiques et d'optimisation tels que les réseaux de neurones, méthodes de régression linéaire / non linéaire et la théorie des probabilités, entre autres.

La (3) mobilité intelligente des chauffeurs de taxi est un facteur clé pour maximiser à la fois la fiabilité du service et sa rentabilité. Bien entendu, la connaissance sur où et quand la demande pour les services de taxi émergeront ajoute un grand avantage concurrentiel pour le chauffeur - en particulier dans les scénarios où il n'est pas viable, d'un point de vue économique, conduire le véhicule au hasard autour de la ville pour trouver le prochain passager. Le choix de la prochaine place où garer le véhicule pour attendre le prochain service est basé sur quatre variables: (i) le prix espéré d'un service dans chaque place au fil du temps; (ii) la distance, dans le temps ou dans l'espace, entre la position actuelle du taxi et la position des places; (iii) le nombre de taxis déjà positionnés dans les places à ce moment-là ; (iv) la demande en chaque place au fil du temps. A notre connaissance, il n'y a pas de recherche qui considère ces quatre variables simultanément. La valeur de la variable (iii) peut être obtenue directement basée sur la position des véhicules de la flotte en temps réel - néanmoins, les trois variables restantes exigent des modèles de prévision à court terme afin d'estimer sa valeur.

La classification de la demande pour les services de taxi à court terme est un problème complexe. Cette recherche peut être décomposé en (iv) le nombre de passagers (soit un entier positif) et (i) la valeur espérée du prix de ces services (c'est-à-dire, une catégorie de prix). Pour résoudre ce problème, nous proposons une méthode qui utilise l'analyse de séries temporelles ensemble avec des techniques de discrétisation. Cette méthode diffère des existantes parce qu'il apprend progressivement et, de cette façon, il s'adapte de manière appropriée à n'importe quel scénario de la demande (c'est à dire des pointes de demande inattendues).

La variable (ii) se rapporte à la quantité de temps nécessaire pour atteindre un point dans la ville / place où il y a des conditions favorables de la demande (par exemple, une forte demande ou une demande de rentabilité élevée). Par conséquence, ce problème est une prédiction du temps de voyage a priori.

La prévision des temps de voyage est un problème amplement étudié dans la littérature. Toutefois, on a effectué une approche à ce problème plus fondamentale que la simple application d'un algorithme d'apprentissage supervisé existant. Cette décision est basée sur deux facteurs: (ii-1) créer une manière durable pour extraire autant d'informations que possible des données envoyées par ces réseaux d'opérateurs, indépendamment de la connaissance de type (c. variable objectif) que nous voulons extraire; (ii-2) être possible d'inclure plusieurs sources de données pour être en mesure d'améliorer la quantité d'informations à la disposition des modèles d'apprentissage. Pour résoudre ce problème, dans ce contexte, il est proposé une nouvelle méthode pour maintenir des statistiques suffisantes concernant une ou plusieurs variables d'intérêt dans une matrice source-destination dynamique dans le temps. Cette nouvelle méthodologie comprend des techniques d'agglomération de données géo-spatiales et des algorithmes d'apprentissage automatique incrémentales.

Tous ces problèmes ont été résolus en utilisant des données réelles confiées par les deux plus grands opérateurs de transport routier public à Porto (Portugal). Ces méthodologies ont obtenu des résultats encourageants dans les expériences qui ont été menées en utilisant ces données, et, de cette façon, ont démontré son utilité dans le monde réel. Ce travail a abouti à seize articles lors de conférences ou publiés dans des revues internationalement reconnues.

Agradecimentos

Esta tese foi realizada entre Setembro de 2009 e Dezembro de 2014. Durante estes mais de cinco anos, muitas foram as pessoas que marcaram o meu percurso pessoal e profissional. Quero começar por agradecer ao prof. João Mendes-Moreira. Por acreditar em mim em todos os momentos. Pelo extraordinário esforço que fez por acompanhar de perto todos os meus passos. Pela forma como me ensinou a escrever trabalhos científicos. E pela espantosa humildade com que o fez. É um investigador de excelência. E no final de contas e talvez o mais importante, um amigo.

Quero agradecer ao prof. João Gama, o mais brilhante investigador com que tive o prazer de trabalhar. Os problemas mais complexos parecem simples para ele...são coisas naturais. A excelência que me exigiu ao longo deste trabalho é fundamental para o nível de qualidade que ele tem hoje. A ele agradeço a oportunidade que me deu de mostrar o meu valor.

Quero também agradecer aos profs. Jorge Freire de Sousa e Michel Ferreira, pelos casos de estudo reais em que me possibilitaram trabalhar, pelo acompanhamento que deram ao meu trabalho e pelas fontes de financiamento que providenciaram para suportar este doutoramento. Termino ainda com uma menção ao profs. Fernando Nunes Ferreira e Augusto Sousa, pelas oportunidades que me deram de iniciar a minha carreira académica como Monitor e Prof. Assistente, respectivamente, e ao prof. Pavel Brazdil pelo exemplo de simplicidade que sempre me transmitiu.

Agradeço ao LIAAD-INESC TEC pelo apoio logístico e financeiro que deu para a realização desta tese. À Geolink, Lda. e à STCP - *Servicos de Transportes Colectivos do Porto* agradeço os dados disponibilizados para realizar este estudo. Ao ”gangue do lanche” pelo companheirismo. Ao Diogo, por ser o meu irmão mais velho. E ao Miguel, por ser o meu irmão ”caçula”. Ao Dumbo, por ser teimoso mas pontual. Ao Rui Costa, pelas conversas à porta do meu prédio. À Márcia, pelas torradas do Itaipu. Ao Hugo, pela aquela cerveja em Moscovo. Ao Daniel, pelas francesinhas. Ao Cardoso, por me dar boleia à Polícia. À PC, pelas cortinas do ET. Ao Ricardo Rodrigues, pelos cabides que arrumou. Ao André, por aquele fim-de-semana em Sofia. Ao Rui Pinto, pela casa para o Poker. Ao Leo, pela Âncora dos Açores. Ao Ruas, por aquele trabalho de grupo em que eu não fiz nada. Ao Oded, pela paciência. À Pri, pela cumplicidade. À Tuna de Engenharia da Universidade do Porto, pelas serenatas. E a todos os outros que não deviam aqui mas que não estão por falta de espaço ou porque eu como muito queijo. Ah! Ao queijo (bem lembrado)! E ao vinho do Pipa

Velha! Às marotices do Garrido. Aos Aeroportos (onde escrevi muito...) e aos aviões! Aos comboios (bielo)russos! E Aos dados de Vegas. Ao gado bravo! Às chuvadas da Tailândia. Aos McDonalds da Finlândia. À Academia Pedro Sousa. Aos sorrisos. À kizomba e aos kizombeiros. Às noites que não têm fim. Aos Jogos Olímpicos. À minha família.

Este trabalho foi apoiado pelo financiamento providenciado pelos projectos DRIVE-IN: "Distributed Routing and Infotainment through Vehicular Internetworking", MISC: "Massive Information Scavenging with Intelligent Transportation Systems", VTL: "Virtual Traffic Lights", KDUS: "Knowledge Discovery from Ubiquitous Data Streams", I-CITY - "ICT for Future Mobility" and MAESTRA com as bolsas CMU-PT/NGN/0052/2008, MITPT/ITS-ITS/0059/2008, PTDC/EIA-CCO/118114/2010, PTDC/EIA-EIA/098355/2008, NORTE-07- 0124-FEDER-000064, ICT-2013-612944, respectivamente.
Quero ainda agradecer o financiamento providenciado pelo ERDF - European Regional Development Fund atravrés do Programa COMPETE (Programa operacional) e pela FCT (Fundação Portuguesa para a Ciência e Tecnologia) atravrés do projecto FCOMP-01-0124-FEDER-022701.

Contents

List of Figures

List of Tables

Part I

The Problem

Chapter 1

Introduction

Nowadays, there are about 800 million vehicles running on our road networks [Ferreira *et al.*, 2012]. These vehicular networks are crucial for human mobility, regardless of their type. The excessive number of vehicles running on the world's biggest urban areas are increasingly discouraging the use of private transportation vehicles in favor of public transportation.

Such large number of vehicles running worldwide is increasing the complexity of the transportation networks, especially its operations. Therefore, it is becoming harder to maintain the efficiency of private transportation. These inefficiencies lead to road congestion, higher levels of pollution, time and energy wastes. Moreover, the increasing price of fuel is turning private transportation into a luxury as the cars' rising operational costs go up to levels which are unfordable in most family budgets. For instance, the congestion indexes in the USA's urban road networks in 2011 caused a total 5.5 billion hours in travel delays and 2.9 billion gallons of fuel wasted [Schrank *et al.*, 2012].

In the last decades, public road transportation companies played a central role in highly populated urban areas, especially by providing fast short distance transportation services. Inner-city transportation networks are becoming larger to cover the increasing demand for fast and reliable transportation to and from their industrial, commercial and residential cores. New challenges await this industry: the aforementioned increase in fuel costs, its effects on ticket's pricing, and the improved offer on railway-based services is forcing the CO^2 market to be more reliable than before so that they can maintain their profitability. In the US, the savings on congestion costs caused by public transportation services increased nearly 131% from 1982 to 2005. However, this increase was 10% between 2005-2011 [Schrank *et al.*, 2012].

Consequently, monitoring their operations is now more relevant than ever. This thesis is focused on taking advantage of the **GPS (Global Positioning System)** to sense each vehicle position. The computer-based applications of this data to provide innovative services related with different modes of transportation are denominated as **Intelligent Transportation Systems (ITS)** [European Parliament, 2010]. However, it is important to retain that ITS can have a larger scope than this by their different domain applications such as urban infrastructures [Ferreira *et al.*, 2010; Nunes *et al.*, 2012], traffic management [Wang, 2005] or even on the different types of data employed on such tasks (e.g.

Loop Detectors [van Lint and Van der Zijpp, 2003]). From now on, we focus on GPS-based ITS focused on public road transportation services.

GPS provided an unprecedented opportunity to develop large mobility networks. It constitutes an *unbounded* stream of data that arrives at a high rate. In the last decades, **Machine Learning** (ML) research have been essentially focused on *batch learning* usually using relatively small datasets. On *batch learning*, the training data is **assumed** to be entirely available to the learning algorithm. It outputs a decision model after processing the data multiple times. By doing so, it is expected to uncover patterns that can somehow help to optimize these networks.

One of the limitations of such batch learning methods is their inability to change their models after the training stage (i.e. in real-time). Hence, the vehicular network's nature is, by definition, highly **dynamic**. Such nature is easily illustrated by sporadic traffic jams that may occur under particular contexts (e.g. the late afternoon of a particular road during a raining Friday when there are works on another road directly connected to this one), which were not within the training dataset. Consequently, these traditional ML algorithms do not fully exploit the information within such location-based data stream. Such task requires knowledge discovery methods able to output values *while* the data is being collected (i.e. **incremental**), to react to unexpected situations such as traffic jams or high demand events (i.e. **concept drift**) by adapting their learning models to the current system status.

The aim of this thesis is to monitor the operations of these vehicular networks to **infer** useful information about their **future status** on both short-term and long-term horizons. To do it so, we start by reviewing the current State-of-the-Art on data driven methods to improve the Operational Planning and Control on Public Transportation (PT) Networks. The aim with this step is to identify research opportunities provided by the GPS-based data which are not properly covered by the existing literature. Finally, we intend to explore both offline and online ML algorithms to provide advances on such topics. As an high-level goal, we expect to increase the profitability of public road transportation companies by **mining** this rich data source.

This Chapter begins by introducing a problem overview in Section 1.1. Then, in Section 1.2 the objectives of this thesis are presented. Finally, the structure of this document is detailed in Section 1.3.

1.1 Problem Overview

The GPS data comprises an accurate and yet cheap source of spatiotemporal information. The basic structure of each sample consists on four variables: (1,2) two coordinates corresponding to the vehicle's latitude and longitude, (3) the vehicle status (e.g. engine on/off) and (4) a timestamp (e.g. Julian). An illustrative example is displayed in Fig. 1.1.

On a first glance, it may seem quite *simple* to infer the future values of these traces (e.g. to plot them over a geographic map to predict the vehicle's destination). However, these vehicular networks provide hundreds of traces like this one in real-time (i.e. one per each vehicle) whose values are **dependent** from each other. If we consider to mine the entire urban dynamics, we may also consider to use additional GPS data sources such as other fleets (e.g. cargo transportation) or even smartphones (see, for instance, [Do and Gatica-Perez, 2014]). Consequently, automatic techniques to uncover the patterns on such traces are needed.

Many PT networks around the world already have these GPS devices in place (e.g. mass transit agencies in New Jersey, Chicago, Minneapolis and Seattle (USA); Ottawa and Montreal (Canada); Eindhoven and The Hague (Netherlands) [Furth *et al.*, 2003]). There are also multiple ITS that already explore successfully this data for Intelligent Routing [Zhu *et al.*, 2010], Bus Travel Time prediction [Chien *et al.*, 2002] or efficient taxi dispatching [Lee *et al.*, 2007]. Typically, such ITS frameworks employ (i) simple descriptive statistics (e.g. moving averages, standard deviations) or (ii) offline ML methods to extract information from the GPS traces. However, these traces comprise an *unbounded* stream of data. This kind of data is produced continuously at a high speed from multiple locations and time granularities which content is highly stochastic. This data possess three key characteristics which make the simple application of the abovementioned learning methods (i,ii) inefficient or even useless. They are enumerated as follows [Gama, 2010]:

1. They provide an infinite source of spatiotemporal information where **novel** concepts are being constantly introduced over time (e.g. an Artificial Neural Network (ANN) trained using traffic flows collected on sunny days will not accurately predict their future values on rainy ones);

2. The probability distribution of their values may **evolve** over time (e.g. a continuous but speedy vehicular flow on a highway is heavily decreased by a traffic jam);

3. Its high arrival rate does not allow to perform high demand computational tasks over the data (i.e. optimal *fitting*) to extract useful patterns (e.g. an ANN takes thirty minutes to be *fitted* to data collected between 2pm and 4pm. However, during such period, one million of novel data samples have arrived. Consequently, the obtained ANN is not reliable).

Recently, techniques to **learn from data streams** had an huge development. Its increasing applications on many sensor networks - like our own ones - proved its efficiency to deal with these characteristics [Rodrigues and Gama, 2009]. In this thesis, we intend to explore such techniques over the GPS streams

LAT	LON	STATUS	TIMESTAMP
37.77851	-122.39635	1	1212507397
37.78093	-122.39325	1	1212507337
37.78448	-122.39498	1	1212507277
37.78765	-122.39494	1	1212507217
37.78951	-122.39734	1	1212507159
37.79148	-122.39964	0	1212505532
37.79070	-122.39952	0	1212505469
37.79274	-122.40134	0	1212505409
37.78894	-122.40184	0	1212505354
37.78895	-122.40206	1	1212505346
37.78929	-122.40313	1	1212505290
37.78852	-122.40554	1	1212505230
37.78803	-122.41032	1	1212505110
37.78745	-122.41494	1	1212505048
37.78689	-122.41823	1	1212504990
37.78677	-122.41819	0	1212504975
37.78511	-122.41797	0	1212504914
37.78317	-122.41735	0	1212504854

Figure 1.1: Illustrative example on a data stream of GPS traces.

of data broadcasted by the vehicular networks comprised in public transportation means. They can be used as complement to offline ML methods or standalone, depending on its final application. The ultimate goal is to produce **sustainable** learning frameworks on a computational view point. Such frameworks must be able to deal with this explosive grow of spatiotemporal data sources in a reliable way. The aplicational focus of such ITS is to improve the operational planning and the real-time control of public road transportation networks. The research question arose on this thesis is defined and discussed below.

1.2 Research Question

In this PhD thesis, we want to test the validity of the following hypothesis:

> *Is it possible to improve the operational planning and control in Public Transportation Networks by mining their location-based streaming data using Machine Learning frameworks?*

The scope associated with this hypothesis is wide due to the high number of problems that can be fitted within. Obviously, we do not intend to solve all in this thesis. The main idea behind testing such hypothesis is to study the State-of-the-Art on these problems in order to identifya very small subset which provide an opportunity to perform significant contributions. By doing so, we intend to propose novel ways to deal with such large amounts of data which can not only provide contributions for these industries but also to open novel research lines on a mid-term future. Such concrete opportunities - as well as the consequent research goals - are introduced throughout Chapter 2.

This hypothesis will be explored using location based-data from public road transportation networks, namely, 1) the buses and 2) the taxis ones. These two networks have some common characteristics and synergies - which corresponds to the **human activities in urban environments**. Such synergies suggest that these networks can be studied together in order to achieve a better comprehension of the phenomenons that affect their behavior They are enumerated as follows:

1. Both are **equipped** with GPS devices which broadcast data **available** to perform this study;

2. Both provide a **continuous stream of data** about the networks behavior based not only on the **vehicles' location** but also on other status variables such as the number of passengers traveling within or the vehicles' mechanical status.

3. Both enclose **vehicular networks** whose operations rely on a) **dependences** and/or b) **correlations** between the vehicles behavior. Some examples of those could be a) the delay propagation effect on a given high frequent bus route introduced by one vehicle failing the schedule or b) the expected distance of a taxi service departing from a location of interest given that the last N vehicles departed from such spot experienced a cruising distance larger (or inferior) to a given time threshold.

4. The **passenger demand** on the transportation services provided by such networks is highly dependent on the **regularities of the human behavior** such as the sleep period at night or the difference between the travelers origins and destinations during workdays/weekends (e.g. the number of night-time boardings of some bus lines on downtown is highly correlated with the pick-up quantity on the nearby taxi stands).

5. The long-term **planning** of these networks is highly dependent on the **seasonality exhibited during the year** such as the scholar holiday period, the Christmas/Easter time or the Summer time. Important planning stages on those networks (e.g.: the location of taxi stands on a given urban area or the definition of the bus schedule plan) are relevant examples of the dependency in place.

6. The **real-time control** of both is highly sensitive to **anomalous demand events** that may unbalance the expected relationship between the service offer and demand - and thereby provoking **unexpected disruptions** on such service. Examples on this issue could be overcrowded buses caused by large scale shows (e.g. sports games, music concerts) or even the first Autumn rains effects, which may cause a temporary absence of taxi offers on some location due to an exponential increase on passenger demand (specially if they are not wearing properly to face such weather condition).

The aforementioned characteristics represent similarities that are reflected on the **data** provided by these networks, namely, 1) by exhibiting the same granularity of the existent regularities (daily, weekly) and 2) highly correlated passenger origin/destination matrices; 3) by revealing the existence of anomalous demand events (allowing thereby their detection in both time and space) or 4) even common data distributions on the time series of passenger counts. Consequently, such streaming data provide challenging opportunities to improve both networks operational planning and control by exploring methods to learn and therefore identify these patterns. An overview on the State-of-the-Art on AVL-based Intelligent Transportation Systems is presented in Chapter 2. By surveying this subject, we expect to identify such opportunities by revealing problems where such approaches may present a contribution to the existent literature. As consequence, a summary of specific research goals on this topic are presented in Section 2.4.

1.3 thesis Structure

The remainder of this thesis is structured as follows:

- Part I: The Problem
  - Chapter 2 comprises an overview on Intelligent Transportation Systems focused on improving the PT Planning and Control using streaming location data. We start by reviewing applications focused on improving the Planning on Mass Transit agencies, namely, on evaluating/improving their Schedule Plan (SP) reliability. Secondly, methodologies to build automatic control strategies are briefly surveyed.

Thirdly, the State-of-the-Art on improving the real-time Control of Taxi Networks is presented. Then, some research opportunities are pointed out from those analysis. Finally, the concrete research goals of these thesis are pointed out. The methodologies proposed to accomplish such goals are presented on the Parts II and III of this thesis.

  – Chapter 3 presents an overview on basic methods to learn from data streams such as algorithms to build incremental histograms, time series analysis techniques and ensemble frameworks.

- Part II: Mass Transit Agencies

  – Chapter 4 details a framework to validate the day coverage of bus schedule plans by mining its historical GPS data. The aim with this method is to discover which are the days in the year that should be included in the same schedule. We did so by (1) firstly extracting the running times from Automatic Vehicle Location (AVL) data of just one route. Then, they are clustered to obtain the optimal day coverage for this specific route. This is done for each route of the network. Secondly, (2) the schedule coverage of each route is assembled to create a consensual cluster that is common to every route in the network, using consensual clustering techniques. Finally, (3) the understandable rules are extracted, obtaining a new schedule plan day coverage.

  – Chapter 5 introduces an Automatic Control framework to avoid Bus Bunching (BB) occurrences in real-time. This framework relies on seven different steps: firstly, (1) we perform long term Link Travel Time Prediction for every future trips on a daily basis using offline regression. These predictions (2) are updated throughout the day using a first order update scheme based on the offline regression residuals. Then, (3) the updated predictions are used together with the recent prediction's residuals to estimate an approximation to headway-based probability distribution on each stop. These distributions (4) are used to compute the BB likelihood on each stop. On other hand, these likelihoods are updated accordingly the prediction updates performed on the step 2. The likelihood of a BB event to occur on the downstream stops is given by (5) a linear combination of the current value of these likelihoods (i.e. a BB score). This score triggers an event alarm every time that it goes above a frequency-based threshold. Finally, these likelihoods are also used (6) to select and (7) deploy a corrective action to avoid such BB occurrences without any human intervention.

- Part III: Urban Mobility

  – Chapter 6 presents a model to predict with a short periodicity the (1) number of services that will emerge at multiple taxi stands/city areas as well as its (2) fare-based type. We did so by employing incremental techniques to maintain histograms of taxi services on different granularities. Then, we used time-varying Poisson techniques ensembled with time series analysis models which are able to learn and predict the underlying passenger demand model in real-time. Finally, we decompose such model into a given number of bins of an histogram representative of the short-term fare probability distribution function of each city area/stand.

– Chapter 7 proposes a novel three-step incremental framework to maintain statistics on the mobility dynamics over a time-evolving origin-destination (O-D) matrix. (i) Half-Space trees are used to divide the city area into dense subregions of equal mass. The uncovered regions are then used to form a quadratic O-D matrix which can be updated by transforming the trees' leaves into conditional nodes (and vice-versa). The (ii) Partioning Incremental Algorithm is then employed to discretize the target variable's historical values on each matrix cell. Finally, a (iii) dimensional hierarchy is defined to discretize the domains of the independent variables depending on the cell's samples. This framework is then applied to perform apriori Travel Time Estimation on the context of a Taxi Network.

- Part IV: Concluding Remarks

  – Chapter 8 concludes this thesis summarizing the work done and describing how the initial goals were accomplished. Finally, ideas for future research are pointed out.

Chapter 2

An Overview on Location-based Data Driven Methods for Planning and Control on Public Transportation Networks

The Global Positioning System (GPS) is a satellite based navigation system developed by the US Department of Defense in 1960 [Theiss *et al.*, 2005]. The system was firstly designed for military purposes to provide navigational fixed data on an hourly basis; however, it did not represent a reliable data source. In 1993, the system became operational and available to civilians. However, its accuracy was low ($> 100m$). It became available massively in 2000 and the position accuracy of a basic GPS increased to $\sim 10m$. This technology rapidly became a standard to obtain real-time information not only in professional transportation fleets, but also in individual vehicles.

Various mass transit agencies have now their fleets equipped with Automated Data Collection systems in order to *track* the vehicles' behavior during their operation. The deployment of such systems usually consists of equipping the fleet's vehicles with (i) a GPS receiver, (ii) sensors of interest and (iii) a communication device of communicating with a remote server. The information obtained is then uploaded to the control center. A simple implementation of this system is displayed in Fig. 2.1. While the GPS receiver is mainly used to track the vehicle's spatial coordinates, other sensors may be employed to collect other types of data such as the (1) stop and start moment, (2) mechanical flags, (3) control messages containing corrective actions to apply to the route or (4) farebox transactions.

The recent large-scale development of this vehicular location data source led to an explosive grew of the **ITS - Intelligent Transportation Systems**. The ITS are advanced applications which aim to provide innovative services related with different modes of transport and enable the users to be better informed and to make a *smarter* use of transport networks [Calabrese *et al.*, 2011]. In general, an ITS relies on location-based information: it monitors and processes the location of a certain number of vehicles to obtain information on estimated travel time, traffic flow and/or incidents, for instance, by monitoring a relatively large number vehicles, it is possible to anticipate an early stage bottleneck and to route the traffic away from it using an alternative road way. One of the main trends of ITS is the improvement of the operational planning and the real-time control of public road transportation networks [Ge *et al.*, 2010; Ferreira *et al.*, 2012; Mendes-Moreira *et al.*, 2012; Li *et al.*, 2012]. From now on, we will just consider this type of ITS applications.

In this Chapter, we review ITS focused on improving public road transportation vehicular networks for passengers - such as buses and taxis - based on historical GPS data. We do not intend to do an exhaustive survey about every and each work related with this topic but just an overview about the most impactful and well known applications of this kind of data on the these type of networks. Our goal is to identify breakthrough areas and/or research topics where extra value can be added. The remainder of the Chapter is structured as follows. In Section 2.1, we cover the AVL-based ITS which aim to improve both operations and planning of bus networks. The Third Section revises the GPS-based works focused on the Operational Control on Taxi Networks. Fourthly, some challenges and research opportunities on these topics are summarized. Finally, an overview of this State-of-the-Art is presented in Section 2.4, as well as the research goals addressed on this thesis.

2.1 Mass Public Road Transportation Networks

One of the most important aspects on improving the operations of a given mass transit company is to determine *if* and *why* their buses are failing to meet the schedule. In fact, reliability problems are a major concern for both passengers and transit operators. A service that is not on time causes an increased waiting time on stops, uncertainty on travel time (TT), Bus Bunching (BB) (i.e. a

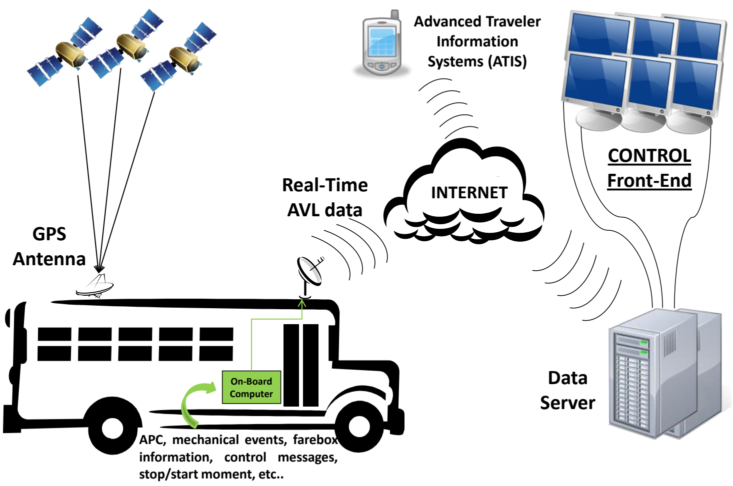


Figure 2.1: Typical Implementation of an Automated Data Collection system.

platoon of two or more vehicles running the same line) events and ultimately, a general dissatisfaction with the system. An unreliable service may lead to a major loss of public support as the passengers may leave these networks to find alternative transportation modes, which leads to a critical loss of revenue [Abkowitz and Tozzi, 1987; Clotfelter, 1993].

After the Tri-Met experience kicked-off in 1991 [Tri-Met, 1991], many companies started to install new computer-aided Bus Dispatch Systems. Examples of cities include New Jersey, Chicago, Minneapolis and Seattle (USA); Ottawa and Montreal (Canada); Eindhoven and The Hague (Netherlands) [Furth *et al.*, 2003]; Cagliari and Genoa (Italy) [University of Southampton (UOS), 2002; Barabino *et al.*, 2013]; Melbourne (Australia) [Mazloumi *et al.*, 2010]; Toulouse (France) [University of Southampton (UOS), 2002] or even London (UK) [Hounsell *et al.*, 2012]. Such dispatch systems were based both on the AVL and on the Automatic Passenger Counting (APC) systems deployed on their fleets. These systems collect the location of buses usually by broadcasting the sensors' values using an interval of 10–30s depending on the radio capacity. Typically, AVL systems are based on GPS measurements while the APC systems typically rely on estimation techniques based on door loop counts or weight sensors. These sensors are installed in every vehicle.

Initially, the service provider only wanted to monitor and control their operations (go to Fig. 2.2 to see a possible example of a monitoring framework). Nevertheless, the advances in real-time communications and vehicle location technologies (such as WiFi, 3G and GPS) over the last two decades largely increased the availability of such data. There has been an increasing evolution from the old asynchronous acquisition methods, where the data acquired in each vehicle were uploaded to a main server with a large periodicity (commonly daily), to a synchronous method (i.e. **real-time**) [Furth *et al.*, 2003]. Such online technology makes it possible to produce continuous flows of data (also known as data streams). Each vehicle transmits the data with a very short (but certain) periodicity to a main server.

In the last decade, many researchers highlighted the potential of the stored AVL data to provide insights on how to evaluate (and improve) Public Transportation (PT) reliability in mass transit companies by improving **Operational Planning and Control**. The technical reports presented by James G. Strathman and his team became the backbone of State-of-the-Art on AVL-based evaluations of schedule reliability [Strathman *et al.*, 1999; Strathman, 2002; Strathman *et al.*, 2003]. However, the real time availability of this AVL data opened new research directions for improving PT reliability, namely by introducing real-time decision models to support the **Operational Control**.

In this Section, we review AVL-based ITS which aim to improve both operations and control of Mass Public Road Transportation Networks. Section 2.1.1 revises fundamental concepts about Operational Planning and Control on this context. The second Section describes the State-of-the-Art on evaluating Schedule Reliability. Section 2.1.3 presents the works focused on improving the Schedule Plan, while Section 2.1.4 revises the research done about real-time control measures in order to maintain on-going trips up to schedule.

2.1.1 Fundamental Concepts on Operational Planning and Control

Often, reliability problems arise in complex public transportation networks with high demand. It is possible to divide the causes of reliability problems into two separate groups: (i) Internal and (ii) External. (i) Internal causes include factors such as driver behavior, passenger boarding and alighting at stops, improper scheduling, route configuration or inter-bus effects, which represent *persistent* problems. (ii) External causes are, by definition, more chaotic and these include traffic congestion and accidents, weather, traffic signs and interferences with on-street parking. The (i) *persistent* problems are addressed using (1) **Operational Planning (OP)** strategies, while the (ii) sporadic problems are mitigated by (2) **Control** strategies. While the (1) OP strategies are often referred to as *preventive* actions which aim to avoid PT unreliability on a long-term perspective, the (2) Control actions have a *corrective* purpose in a very specific and brief moment [Abkowitz and Tozzi, 1987; Fattouche, 2007].

Operational Planning strategies

A typical OP process is carried out by sequentially following four steps [Ceder, 2002; Mendes-Moreira, 2008]:

1. **Network Definition**: It consists of defining the *lines*, *routes* and subsequent bus stops. Here, a *route* is considered a road path between an origin and a destination which passes by multiple bus stops. A *line* is defined as a set of routes (which typically comprises two routes with very similar paths, but inversely ordered).

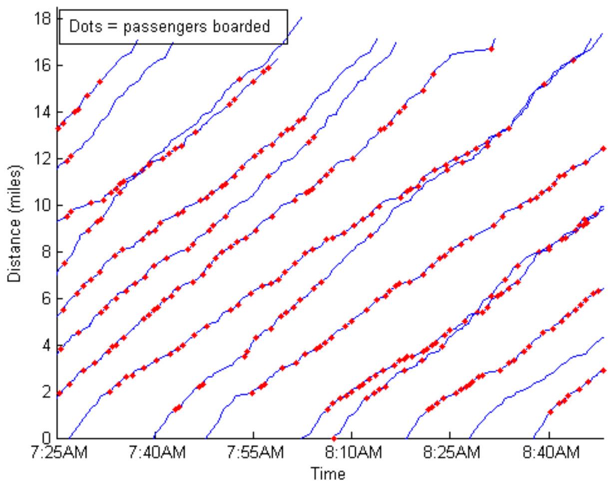


Figure 2.2: Diagram about a real-time visual control framework on a given route at a morning peak hour. The x-axis represents time, and the y-axis is the vehicle location along its route. The red dots represent the bus stops. Image originally from [Berkow *et al.*, 2007]

.

2. **Schedule Planning**: The trip's timetables are defined by firstly identifying the set of bus stops for which schedule time points will be set (the origin/destination stops are always part of this set). Secondly, timestamps are assigned to previously defined schedule time points. Such timestamps may be composed of an expected arrival time plus some slack time. However, in high frequency routes, this timetabling can also be defined by setting the time between two consecutive trips in the same route (i.e., headway-based) [Ceder, 2002]. The set of planned trips is often defined as the **Schedule Plan**.

3. **Definition of Duties** : A duty is a task that a driver and/or a bus must perform. The definition of the drivers' duties has much more constraints than the definition of bus duties (for instance, a driver must stop regularly; governmental legislation). Commonly, the logical definition of bus duties is performed prior to the drivers' duties.

4. **Assignment of Duties**: It consists of physically assigning the previously defined logical duties to the companies' drivers and buses.

The AVL-based OP strategies to improve PT reliability consist into adjusting the definitions made on such tasks using real-world data. This type of works focus on (1) restructuring the route and adjusting the existing (2) Schedule Plan (SP). AVL-based works on this subject follow this trend. The (3) and (4) resource-based strategies are applied to improve the profitability rather than the company's operations. Specifically, the (3) definition and (4) assignment of duties are commonly performed by using constraint-based methods and not by analyzing AVL data. In fact, to the best of our knowledge, there is no work suggesting it so.

Control strategies

It is reasonable to define **Control Strategies** as real-time responses to sporadic service problems [Strathman *et al.*, 2000]. The goal is to restore service normality when deviations occur (i.e., in *real-time*) [Fattouche, 2007]. It is possible to divide these strategies into two different applications [Dessouky *et al.*, 2003]: (i) maintaining schedule reliability using metrics such as on-time performance or headway stability (to be discussed in Section 2.1.2) and (ii) schedule coordination at terminals/hubs to facilitate transfers [Hadas and Ceder, 2010]. This review focuses mainly on the type (i) applications.

Such strategies imply the selection of *corrective* actions (described in Section 2.1.4) to avoid eminent unreliable contexts, which are particularly chaotic in high frequency routes.

2.1.2 On Evaluating Schedule Plan Reliability

The SP reliability is a vital component for service quality. Improvements on reliability may increase the service demand and, consequently, the companies' profitability. Low reliability levels lead to a limited growth in the number of passengers and to a decreased perceived comfort [Strathman *et al.*, 1999]. It is possible to establish three distinct axis on evaluating SP reliability [van Oort,

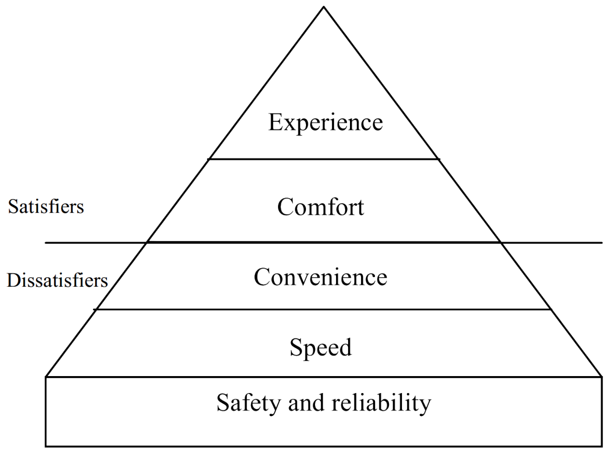


Figure 2.3: PT quality factors presented using a Maslow's pyramid. It defines the boundary on the desirable/essential factors. Image originally from [Peek and van Hagen, 2002].

2011]: (1) the unexpected increases on the waiting time on bus stops; (2) the time spent in crowded situations caused by transport overloading; and (3) delays on the passengers arrivals due to Travel Time Variability (TTV). The first two (1-2) axis are mainly related with passengers *comfort* and *experience* criteria. The value of such extra time consumptions vary from the passenger condition (seated or standing) [Wardman and Whelan, 2011]. However, these two aspects are mainly **satisfiers**: additional aspects that the passengers like to have but are not essential factors to abandon the services provided by a certain PT company. On the other hand, the last one (3) is a fundamental issue by the disturbances that it does on the passengers daily activities [van Oort, 2011]. By affecting directly the *convenience* and the *speed* of transportation, it is key to maintain the travelers confidence on the PT network (i.e. a **dissatisfier**). These priorities on the described PT quality factors are illustrated in Fig. 2.3. Once established, it is expected that a SP meets the passengers' demand by *following* their mobility needs (namely, their daily routines). Typically, service unreliability is originated by one (or many) of the following causes [Fattouche, 2007; Cham, 2006]: schedule deviations at the terminals, passenger load variability, running time variability, meteorological factors and driver behavior.

Nowadays, urban areas are characterized by a constant evolution of road networks, services provided and location (for instance, new commercial and/or leisure facilities). Therefore, it is highly important to automatically assess how the Schedule Plan suits the needs of an urban area. An efficient evaluation can lead to important changes in a SP. These changes will lead to: a reduction in operational costs (for instance, by reducing the number of daily trips in a given route) and/or a reliability improvement in the entire transportation network, which will increase the quality of the passengers' experience and, therefore, the number of costumers.

A SP consists of a set $\mathcal{S} = \{S_1, S_2, ..., S_k\}$ of k schedules which provide detailed information about every trip running on previously defined routes. Each schedule contains a timetable $t_i : i \in \{1, ..., k\}$. Different routes may have different timetables. Nevertheless, they share the number k of schedules and the daily **coverage** C_i of each schedule.

A Schedule Planning process for a given route relies on three distinct steps: (i) the first step is defining the number k of schedules and their individual coverage, C_i. Secondly, (ii) the *schedule timepoints* are chosen among all bus stops in the route, and finally, the third step (iii) is defining a timetable t_i for each route schedule S_i containing the time the buses pass at each scheduled timepoint (per trip). This process is done for all routes. It should be guaranteed that the number k of schedules and the coverage C_i are the same for all routes so that the passengers easily memorize the SP. To learn more about this topic, the reader should see the survey on Urban Transit Operational Planning by Ceder [2002].

From the above mentioned definition of SP, it is possible to divide the SP evaluation into two different dimensions: the suitability of the number of schedules k and of the set of their daily coverages $\mathcal{C} = \{C_1, C_2, ..., C_k\}$, and the reliability of their timetables $\{t_1, ..., t_k\}$ (to test whether the real arrival times of each vehicle at each bus stop are meeting the previously defined timetable). Although there is an obvious impact on the definition of the timetable, to the best of our knowledge there is no research in the literature addressing the evaluation of the number of schedules and their daily coverage.

This Section defines and reviews evaluation methodologies regarding the reliability of timetables.

Evaluation Metrics and Requirements

Firstly, the metrics related to axis (3) are reviewed, as well as the studies that have employed them. Then, two metrics on the evaluation axis (1-2) are presented (which were less used by the existing research on AVL-based PT evaluations).

When evaluating a SP, it is important to differentiate low and high frequency services [Turnquist, 1982]: in *low frequency services*, passengers arrive at the bus stops shortly before the bus's scheduled services, while in *high frequency services* the customers tend to arrive at the stops *randomly* [Ceder and Marguier, 1985]. In the first scenario, *punctuality* is the main metric, while the service *regularity* is the most important metric in high frequency routes. There is no exact boundary between these two scenarios. Recent studies used 10-12 minutes as a threshold between low/high frequency services [Trompet *et al.*, 2011; van Oort, 2011].

Four main indicators were firstly proposed by Nakanishi [1997] and followed by other similar studies [Strathman *et al.*, 1999; Barabino *et al.*, 2013]. These indicators are outlined as follows: (i) **On-Time Performance**, (ii) **Run Time Variation**, (iii) **Headway Variation** and (iv) **Excess Waiting Time**. The first two indicators (i,ii) are more applicable to low frequency routes, while the last two (iii,iv) focus on the high frequency routes [Turnquist, 1982; Strathman, 1998; Strathman *et al.*, 1999]. This set of indicators are the most widely used and they are presented below.

(i) **On-Time Performance** (OTP) indicates the probability that buses will be where the schedule says they are supposed to be. It is possible to represent this metric by an *Arrival Delay* (AD) in a given trip i, i.e., AD_i as function of both the *Scheduled Arrival Time* (SAT_i) and the *Actual Arrival Time* (AAT_i). Therefore, it can be defined as follows [Strathman, 1998]:

$$AD_i = AAT_i - SAT_i \tag{2.1}$$

The (ii) **Run Time Variation** (RTV) represents the variation on the run times performed by each trip. Some introductory concepts on this subject will be presented below. Typically, the TT reports the trip duration, from terminal to terminal, and is often referred to as *round-trip time* [Mendes-Moreira, 2008]. TT is often used to define the time required to go from one point of interest to the other [Turner *et al.*, 1998]. This last definition is used in this review. One of the factors that mostly affects the RTV is the *dwell time*, which is the total time the bus has to stay at a given bus stop for passenger boarding and alighting [Transportation Research Board, 2003].

From the passenger perspective, a larger variation can mean a longer waiting time in some stops and/or missed transfers. From the operational planners' perspective, greater RTV translates into higher costs as a result of the extra hours that must be added to accommodate passenger load variation [Strathman, 1998]. This indicator is more appropriate for routes that cover long distances, facing many traffic lights and regular traffic delays [Sterman and Schofer, 1976].

Given a set of n trips of interest, it is possible to compute the RTV as follows [Strathman *et al.*, 1999]:

$$RTV = n^{-1} \times \sum\nolimits_{i=1}^{n} |SAT_i - AAT_i| \,/\, AAT_i \tag{2.2}$$

In highly frequency routes, where the trips start within very short headways, the on-time performance is not that relevant [Hounsell and McLeod, 1998]. The (iii) **Headway Variation (HV)** represents the probability that controllers are able to maintain a regular and stable headway between each pair of vehicles running in the same routes.

Let $f_{i,j}$ be the frequency (i.e., scheduled headway) established between a given pair of trips, (i, j), while $H^b_{i,j}$ represents the observed headway on such pair of trips at a bus stop of interest, b. The *Headway Ratio* on the bus stop b, i.e., $Hr^b_{i,j}$ is defined as follows [Strathman, 1998; Strathman *et al.*, 1999]:

$$Hr^b_{i,j} = (H^b_{i,j}/f^b_{i,j}) * 100 \tag{2.3}$$

where the value 100 represents a perfect SP matching. Given a set of n trips of interest, it is possible to compute the Standard Deviation and the Mean value of Hr (σ^b_{Hr} and μ^b_{Hr}, respectively). We can do it so by calculating every possible $Hr_{i,i+1} : i \in \{1, ..., n-1\}$ at a bus stop b. Then, it is possible to obtain the (HV) at bus stop b throughout these n trips as follows [Lesley, 1975]:

$$HV^b = \sigma^b_{Hr}/\mu^b_{Hr} \tag{2.4}$$

The (iv) **Excess Waiting Time** is an estimation of the excessive waiting time that passengers experience as a consequence of unreliable service. It is possible to calculate the *Excess Waiting Time* at a bus stop b, i.e., EWT^b as a function of HV_b. A possible way to do so is presented as follows [Welding, 1957]:

$$EWT^b = \frac{{\sigma^b_{Hr}}^2}{2 \times \mu^b_{Hr}} \tag{2.5}$$

The bus stop b used to compute statistics on the first two indicators (i,ii) is the destination bus stop. For the last two indicators (iii,iv), any bus stop can be considered as reference if it has a frequency scheduled to it, i.e., $f^b_{i,j}$. Commonly, such statistics are computed by the transit companies aggregating its values to a fixed time granularity (typically, one hour periods) [Barabino *et al.*, 2013], but they can also be computed according to the trip.

An irregular service implies an unexpected increase in passengers' waiting times - which is naturally more related with the first two axis (1,2). This kind of unreliability could be measured in terms of **Average Waiting Time (AWT)** [Cats *et al.*, 2010]. Let $PAV^{b_j}_{z,k}$ be the arrival time of a given passenger z to a bus stop b_j of a given route immediately before the vehicle performing trip k arrives to b_j. Then, it is possible to compute AWT of a route with s bus stops and T planned trips as follows

$$AWT = \frac{1}{\mathbb{B}} \sum\nolimits_{k=1}^{T} \sum\nolimits_{z=1}^{B^k} \sum\nolimits_{i=1}^{s} AAT^{b_i}_k - PAV^{b_i}_{z,k} \tag{2.6}$$

$$\mathbb{B} = \sum\nolimits_{k=1}^{T} B^k; B^k = \sum\nolimits_{i=1}^{s} bo^i_k \tag{2.7}$$

where B^k stands for the total number of passenger boardings on a given trip k and bo^i_k denotes the number of boardings on a given bus stop b_i on trip k. This metric allows to directly assess the impact of alternative operational control measures on passengers waiting times. However, such measures (e.g. holding a bus in order to coordinate transfers) may also induce longer in-vehicle times [Cats *et al.*, 2010]. In order to evaluate the overall impact on passenger travel times, the **Average In-Vehicle Time (AIVT)** should also be considered. Let $bs_{z,k}$ be the *boarding stop* of a passenger z on a trip k and $as_{z,k}$ the alighting one on the same trip. The $AIVT$ can be computed as

$$AIVT = \frac{1}{\mathbb{B}} \sum_{k=1}^{T} \sum_{z=1}^{B^k} \left(AAT^{as_{z,k}}_k - AAT^{bs_{z,k}}_k \right) \tag{2.8}$$

The AWT and the AIVT are relevant metrics when it comes to evaluate the service convenience on the passengers perspective. However, the State-of-the-Art on AVL-based evaluations do not account that much on these indicators. These evaluations were mainly done on the company's perspective by considering just metrics addressing how well the network behavior fits to the SP. The Section below presents a review of the evaluation of SP reliability by measuring these indicators (i.e. axis (3)) on historical AVL data.

A Review on SP Evaluation Studies

Many works have evaluated schedule reliability by measuring the aforementioned indicators on historical AVL data sets. Strathman [1998]; Strathman *et al.* [1999] evaluated schedule reliability on the Tri-Met by measuring indicators (i-iv), while the work by Bertini and El-Geneidy [2003a] solely focuses on the first two ratios. Traditionally, the HV was often disregarded by the transit planners due to the intrinsic *chaos* assumed (as the schedule timepoints on the timetables are not the central variable to confirm service reliability). Nevertheless, recent advances have changed this reality: in [Strathman *et al.*, 2003] AVL/APC data was considered to evaluate the impact of the HV on the operational control. Another perspective of the Tri-Met data is presented in [Berkow *et al.*, 2007], where an analysis of indicators (ii-iv) demonstrated the feasibility of using AVL data along with other data sources to better accomplish their evaluation. Lin and Ruan [2009] formulated a probability-based headway regularity metric (HV). Then, the authors tested their approach using AVL data from Chicago. In [Bellei and Gkoumas, 2010], relations between transit assignment, BB events and operation models are mined from the location-based data. This study aimed to identify irregularities in HV's distribution function caused by an inadequate schedule plan. The reliability of an express service implemented in Montreal, Canada, is evaluated in [El-Geneidy and Surprenant-Legault, 2010] by employing the indicators (i) and (ii). A large-scale evaluation was performed by Hounsell *et al.* [2012], where the data acquired through the *iBus* (an AVL/APC framework installed on a bus fleet running in the city of London, United Kingdom) was used to evaluate all the four main indicators of schedule reliability.

Another approach to evaluate schedule reliability on a route is the segment-based one. It consists of identifying segments/parts of a route where there are greater schedule deviations and, therefore, the SP should be adjusted by changing the timetable or by introducing bus priority lanes and/or traffic signals in intersections. One of the first authors to realize such work was Horbury [1999] based on the HV. In [Mandelzys and Hellinga, 2010], it is proposed to measure indicators (i-ii) using stop-based metrics, and to identify the causes for larger deviations through an empirical framework. The work in [El-Geneidy *et al.*, 2011] proposes a way of identifying where the schedule is unreliable by evaluating the first two indicators on the schedule timepoints.

Recently, the methodological approach to evaluate SP reliability has evolved from the key indicators to using non-parametric deterministic methods such as Data Envelopment Analysis (DEA) [Mendes-Moreira and Sousa, 2014]. The main advantage enabled by employing such a complex method is the possibility of directly comparing metrics from distinct dimensions by introducing decision-making units. Lin *et al.* [2008] used AVL data to establish confidence intervals for the DEA scores based on the four indicators previously introduced. Despite its usefulness in identifying cost-based relationships between the resources used and the service produced, the DEA models are not addressed in this review as they usually address an wider scope on the companies' management than our own. A comparative overview of the aforementioned studies on evaluating SP reliability is presented in Table 2.1.

The four metrics are well established in the literature. However, they focus mainly on the passengers' perception of service quality, especially the (iii) HV and the (iv) EWT. The (i) OTP can help the planners to identify the *exact* schedule timepoints to be changed, while the (ii) RTV shows a more general perspective on network service, which can lead to more profound studies on the drivers' behavior, terminal dispatching policies or on the current schedule's slack. The (iii) HV is the most used metric. Even so, it is possible to observe that the companies' perspective on such RTV is not addressed as a primary goal of these evaluation studies. Indeed, high frequent routes are usually the main concern of PT planners because those are the ones more sensitive to small deviations. Additionally, they are the ones that have more impact on the public (i.e. more trips and passengers).

Nevertheless, even if it is possible to identify *what* is happening and *where* changes must be performed to improve SP reliability, it is not easy to identify **how** it is possible to improve it. The next Section focuses on the use of AVL data to develop and/or improve Schedule Planning.

2.1.3 On Improving Schedule Planning

Schedule Planning strategies aim to reduce the likelihood of schedule deviations responding to persistent and predictable problems [Fattouche, 2007]. The questions brought about by the researchers when regarding **Schedule Planning** address both the evaluation and improvement of company **timetabling**. The timetable adjustments can be proposed in three perspectives: **(A) slack-time based**, **(B) travel-time based** and **(C) headway based**.

In the real world, there is no perfect SP. The PT operations will *certainly* experience some TTV which will lead to some unreliability comparatively to the previously defined timetables. The aforementioned techniques try to reduce this SP unreliability as much as possible. Typically, (B) the travel-time based strategies to improve the SP consist of changing the scheduled round-trip times. For that, these strategies use some inference method in order to predict such variables. However, any prediction produced has an associated likelihood. Consequently, such prediction values need to be *tuned* before going to the public schedule[1]. One of the most common tuning strategies consists of (A) adding slack times to these predictions (especially in low frequency routes) based on such variability, as suggested by [Jorge *et al.*, 2012]. A distinct strategy is the (C) headway-based ones: they try to establish optimal bus frequencies by computing a balanced relationship between the expected demand and the available resources. This Section presents a systematic revision of these optimizing

[1] The operational timetable may differ from the one distributed to the general public to improve the passengers' perception on the quality of service.

Table 2.1: AVL-based Research on Evaluating the SP Reliability.

Publication	Granularity	Evaluation Indicators			
		OTP (i)	RTV (ii)	EWT (iv)	HV (iii)
([Strathman, 1998; Strathman *et al.*, 1999])	*route-based*	√	√	√	√
([Horbury, 1999])	*segment-based*				√
([Strathman *et al.*, 2003])	*route-based*				√
([Bertini and El-Geneidy, 2003a])	*schedule timepoint*	√	√		
([Berkow *et al.*, 2007])	*schedule timepoint*		√	√	√
([Lin *et al.*, 2008])	*route-based*		√		√
([Lin and Ruan, 2009])	*schedule timepoint*				√
([Mandelzys and Hellinga, 2010])	*bus stop-based*	√	√		
([Mazloumi *et al.*, 2010])	*route-based*		√		
([Bellei and Gkoumas, 2010])	*schedule timepoint*				√
([El-Geneidy and Surprenant-Legault, 2010], [El-Geneidy *et al.*, 2011])	*schedule timepoint*	√	√		
([Jorge *et al.*, 2012])	*route-based*	√			
([Moreira-Matias *et al.*, 2012b])	*bus stop-based*				√
([Hounsell *et al.*, 2012])	*schedule timepoint*	√	√	√	√
([Barabino *et al.*, 2013])	*bus stop-based*				√

strategies for improving the schedule timetables.

(A) Tuning up the Schedule using Slack Times

Operational Planners may add slack times to the timetable when they are building a schedule for a low frequency route. The **slack time** is the difference between the scheduled and the actual *expected* arrival time [Zhao *et al.*, 2006; Yan *et al.*, 2012]. It may be seen as a way to accommodate the *expected* RTV on a schedule timepoint. The amount of slack introduced can produce large scale effects on the SP reliability. Insufficient slack times reduce the likelihood of a bus catching up when it falls behind. On the other hand, excessive slack times will reduce the service frequency or increase the amount of resources required, namely, buses and drivers. The definition of an optimal slack time is a problem that can be expressed as a trade-off function between the service frequency and its reliability. This Section addresses the problem of improving SP reliability by determining an optimal slack time based on AVL data.

It is possible define the optimal slack time as follows [Dessouky *et al.*, 1999; Zhao *et al.*, 2006; Yan *et al.*, 2012]: let $SRT_i^{b_s,b_d}$ be the Scheduled Run Time for a scheduled trip of interest i between the schedule timepoints b_s and b_d, respectively, while $ART_i^{b_s,b_d}$ is the Actual Run Time. It is possible to define both using the following equations:

$$SRT_i^{b_s,b_d} = SAT_i^{b_d} - SDT_i^{b_s}, ART_i^{b_s,b_d} = AAT_i^{b_d} - ADT_i^{b_s} \tag{2.9}$$

where $SAT_i^{b_d}, AAT_i^{b_d}$ are the Scheduled and Actual Arrival Times, respectively, while $SDT_i^{b_d}, ADT_i^{b_d}$ represent the Scheduled and Actual Departure Times between the schedule timepoints b_s and b_d. Then, it is possible to define the Expected Run Time in the same context, i.e., $E(RT)_i^{b_s,b_d}$ as follows:

$$E(RT)_i^{b_s,b_d} = n^{-1} \sum\nolimits_{j=1}^{n} ART_j^{b_s,b_d} \tag{2.10}$$

where n represents the number of previous occurrences of the scheduled trip i (i.e., the same service, day type and Scheduled Departure Time in previous days) considered to compute $E(RT)_i^{b_s,b_d}$. Finally, it is possible to define the optimal slack time to be added to the schedule point b_d in the trip i, i.e., $st_i^{b_d}$ as follows [Zhao *et al.*, 2006; Yan *et al.*, 2012]:

$$st_i^{b_d} = SRT_i^{b_s,b_d} - ({SRT_i^{b_s,b_d}}^2 / E(RT)_i^{b_s,b_d}) \tag{2.11}$$

Although it is common to add slack to the schedule, research focusing on setting appropriate slack times based on historical AVL data is scarce. It is especially surprising if we consider that the slack time is defined according to the *mean* TT, which can be easily computed using AVL data.

To the best of our knowledge, there are just four works employing AVL data to optimize slack times in timetables: Dessouky *et al.* [1999] found an optimal slack ratio of 0.25 to be added to the SP in place on a transit agency operating in Los Angeles, USA. In [Mazloumi *et al.*, 2012], two heuristics are proposed to solve the timetabling problem as an optimization problem by employing two heuristic procedures, Ant Colony [Dorigo and Gambardella, 1997] and Genetic Algorithms [Melanie, 1999]. Both the travel and the slack times were considered output variables. Such methodology was calibrated using location-based data from a bus route in Melbourne, Australia. Yan *et al.* [2012] proposed a novel optimization model to schedule design by taking into account the bus TT uncertainty and the bus drivers' schedule recovery efforts. The goal with this model was to find the optimal slack time to add to the schedule running in the city of Suzhou, China. Finally, a distribution rule-based methodology is proposed in [Jorge *et al.*, 2012] and the goal here is to find particular conditions which lead to schedule unreliability. Then, an optimal slack time equation is formulated based on the probability density function (p.d.f.) found in each unreliable context in the city of Porto, Portugal.

By adding slack time to their schedules, the operational planners expect not only to increase the passengers' satisfaction on service reliability, but also increase the flexibility of both the operators and controllers so as to take actions for the vehicles to recover their scheduled times. Often, the slack also addresses regulatory questions regarding the maximum operator driving time, as well as other terminal bus dispatching issues. Due to its importance for the operations and perception on the service quality, further research should be conducted on this topic based on historical AVL data. Nevertheless, this tuning strategy highly depends on the scheduled arrival times. The next Section presents a comprehensive review of AVL-based research techniques to improve these times.

(B) Travel Time Prediction

One of the most common transportation problems is the Travel Time Prediction (TTP). The literature on this topic is extensive. TTP problems can be used in several contexts such as fleet management, logistics, individual navigation or mass transit planning, monitoring and control. This Section provides a review of the research focusing on TTP based on AVL data to improve public road transportation planning and monitoring.

TTP consists of predicting the TT for a given trip (or segment). The TT function is formally defined below. Let $TT_{(i,j)}$ be the run time between two bus stops of interest $b_i, b_j : j > i$. It is possible to compute TT as follows.

$$TT_{(i,j)} = \sum\nolimits_{k=i}^{j-1} dwT_k + RT_{(k,k+1)} \tag{2.12}$$

where $RT_{(k,k+1)}$ is the non-stop running time in the road segment between two consecutive bus stops b_k, b_{k+1} and dwT_k is the dwell time on the bus stop b_k.

Although some approaches seem quite simple, various methodologies are employed to TTP problems from different research areas. It is possible to divide such approaches into four distinct categories [Chien *et al.*, 2002; Carrascal, 2012]: (B-a) Machine Learning (ML) and Regression Methods, (B-b) State-Based and Time Series models, (B-c) Traffic Theory-based models and (B-d) Historical data-based models. This last family of naive approaches consists of simple averages and other type of time-varying Poisson processes whose average TT or speed is achieved by its historical values depending on the day type and/or on the period of the day [Chung and Shalaby, 2007]. Its simplicity is commonly reported as an important drawback when representing the complex relationships between the TT and other variables usually established in urban public transportation networks. Consequently, they present a poor approach to TTP [Carrascal, 2012; Kieu *et al.*, 2012]. The (B-c) Traffic Theory-based models are

Table 2.2: Complex Regression models employed in TTP problem-solving.

Publication & Denomination	**Description**	**Examples**
Artificial Neural Networks (ANN) [Rosenblatt, 1958]	It uses multiple layers of artificial neurons which, together with the link's weights, are able to establish complex relationships between the input and the output values.	[Chien *et al.*, 2002], [Chen *et al.*, 2004], [Jeong and Rilett, 2005], [Patnaik *et al.*, 2006], [Gurmu, 2010], [Wong, 2011], [Khosravi *et al.*, 2011b,a], [Mazloumi *et al.*, 2011], [Zaki *et al.*, 2013]
Kernel-based Regression [Nadaraya, 1964]	Instance-based learning method that uses a kernel function to assign weights to each training sample accordingly with their similarity to the target one.	[Sinn *et al.*, 2012], [Dessouky *et al.*, 2003]
k-Nearest Neighbors Regression (kNN) [Cover and Hart, 1967]	This method finds the k closest samples in the historical database using some distance metric of interest and combines its outputs (usually by calculating their average).	[Klunder *et al.*, 2007], [Chang *et al.*, 2010a], [Baptista *et al.*, 2012], [Sinn *et al.*, 2012], [Dong *et al.*, 2013]
Projection Pursuit Regression (PPR) [Friedman and Stuetzle, 1981]	The model consists of linearly combining non-linear transformations in the linear combinations of explanatory variables.	[Mendes-Moreira *et al.*, 2012]
LOcally WEighted Scatterplot Smoothing (LOWESS) [Cleveland, 1981]	Non-parametric regression method that combines multiple classical regression methods in a kNN meta-model.	[Vu and Khan, 2010]
Support Vector Regression (SVR) [Drucker *et al.*, 1997]	This method uses a max. threshold ϵ which stands for the residual between the target function and any of the training samples. It is used to establish a hyperplan to define that function which contains all these training samples.	[Bin *et al.*, 2006], [Mendes-Moreira *et al.*, 2012]
Random Forests (RF) [Breiman, 2001]	Random Forests is a bagging-type ensemble method which employs decision tree induction where the split criteria is set using a randomly selected feature subset.	[Mendes-Moreira *et al.*, 2012]

well known for handling traffic management - but not that commonly applied on AVL-based TTP methods for improving the schedule planning [Carrascal, 2012] (to read more about this type of formulations, the reader should go to Section 3.2 in [van Lint, 2004]). For these reasons, just the two first categories (B-a,B-b) are addressed on this review. Table 2.2 presents a detailed description of some complex regression models employed in TTP works.

It is possible to differentiate short- and long-term TTP problems according to the prediction horizon considered. It is common to define such threshold between 60 to 180 minutes [Carrascal, 2012; Kieu *et al.*, 2012]. The long-term TTP is most commonly used for the SP definition - which is the functionally addressed by this review. This is an interesting problem due to the existing amount of historical AVL data in the agency databases used today. To accomplish such goal, the prediction should be valid for a long period (for instance, TTP for Monday trips at 8am should be as accurate as possible for the entire forecasting horizon, i.e., all the days using that planned trip, typically, several months or even years). However, the State-of-the-Art on long-term TTP is nearly non-existent. To the best of our knowledge, there are just two works on it: Klunder *et al.* [2007] uses kNN with the input variable departure time, weekday and date; a comparison of State-of-the-Art regression algorithms (SVR, PPR and RF) for long-term TTP is presented in [Mendes-Moreira *et al.*, 2012].

Almost every TTP approaches consider a short-term horizon. The short-term TTP is commonly related to the real-time information on arrival time provided to the clients by the Advanced Traveler Information Systems (ATIS) in place. It is very useful to passengers as it improves both their traveling experience and their transfers [Kieu *et al.*, 2012]. The techniques employed to solve this kind of problems are necessarily different from the long-term TTP[2] and are not directly applicable to the **Planning** stage. However, there are many synergies and commonalities between these two problems and, consequently, between the approaches used to solve each one of them: (1) Both are regression problems; (2) consequently, the majority of the algorithms that can be used for the first problem can also be applied to the second one; (3) Nowadays, the information provided by the ATIS on the short-term TT may reduce some passenger-centered TTV, namely excessive passenger loading at some bus stops and/or major hub stations. This effect will cause a chain reaction by reducing firstly the ADT_i and consequently the SDT_i and the TT associated with such stops SRT_i (see eq. (2.9)). This ultimate effects of this process are the reduction of the TTV and the consequent increase of the schedule reliability (which are the goals of long-term TTP on this context). As is further discussed in Section 2.3.3, we believe that the rich literature on short-term TTP can present useful lessons to improve the current studies on long-term TTP. For these reasons, the short-term TTP was included in this review's scope. From now on, we will refer to TTP using a short-term horizon.

The remainder of this Section reviews the works using the approaches to TTP from categories (1-3), followed by methods to evaluate the reliability of such numerical predictions.

(B-a) Machine Learning and Regression Methods

These methods are proposed to infer the arrival times (i.e., a dependent variable) using a mathematical function based on a set of independent variables (i.e. decision variables). Over the last two decades, regression models have been the State-of-the-Art on this kind of approach. Works using such type of approaches are summarized in Table 2.2. Besides providing accurate TTP, the regression models are also commonly capable of estimating the impact that each input variable has on the target variable (i.e., TT). A dwell time-based simple Linear Regression model is employed by [Bertini and El-Geneidy, 2003b; Bin *et al.*, 2006; Tan *et al.*, 2008; Tétreault and El-Geneidy, 2010]. However, complex models such as SVR, kNN, PPR and ANN are the most popular approaches to this problem due to their ability to find complex non-linear relationships

[2] Besides the obvious differences in time horizons, the seasonalities detected and the importance of the decision variables are completely different from one problem to the other. The differences have a relevant impact when it comes to finding the relationships between these variables and the target variable.

between the target variable and the independent ones.

ANN is the most successfully regression method employed on TTP problems: [Chien *et al.*, 2002; Jeong and Rilett, 2005; Gurmu, 2010; Wong, 2011] use it over location-based data while the works in [Chen *et al.*, 2004; Patnaik *et al.*, 2006] use APC-data. However, it presents four main drawbacks comparatively to other regression methods: (1) a time consuming training procedure [Bin *et al.*, 2006]; (2) the input-output function is unknown[3]; (3) a reasonable knowledge of the problem is usually required to perform an optimal feature selection, hidden layers and learning rate [Bin *et al.*, 2006], and (4) overfitting is highly possible [Chen *et al.*, 2004].

Approaches promising to mitigate three of these four limitations have recently been presented: (2) in [Mazloumi *et al.*, 2011] a method to perform ANalayze Of VAriance (ANOVA) [Box, 1953] to determine feature selection in order to perform ANN-based TTP; (3) in [Khosravi *et al.*, 2011a], a Genetic Algorithm [Melanie, 1999] is proposed to find the optimal values for the ANN parameters in TTP context; (4) in [Khosravi *et al.*, 2011a; Mazloumi *et al.*, 2011] proposes to find prediction intervals rather than optimal values for TT to handle the uncertainty within the ANN predictive models. Such prediction intervals reduce the possibilities of overfitting and can be used to optimize the schedule's slack times. However, such additions to the basic ANN model decreases even more the (1) traditionally slow training process by adding complex preprocessing stages.

The SVR have the advantage of being able to incorporate different types of kernels to find the optimal boundary [Bin *et al.*, 2006], while kNN and kernel based regression models deal more adequately with missing data or with *outliers*[4].

The AVL-based kNN models for TTP emerged recently [Chang *et al.*, 2010a; Baptista *et al.*, 2012; Sinn *et al.*, 2012; Dong *et al.*, 2013]. Some works report that they can outperform ANN [Sinn *et al.*, 2012; Liu *et al.*, 2012]. Besides the aforementioned characteristics, the kNN is an approach which, conversely to the ANN or the SVR, does not require any assumption about the functional form of the relationship between the dependent variables or the statistical distribution of data (i.e., non-parametric). However, similarly to ANN/SVR methods, its reliability depends on the availability of a *sufficiently* large quantity of data [Liu *et al.*, 2012].

Even though they are useful, most methods reported do not provide a clear input-output function as Linear Regression models do. Surprisingly, there are not many works comparing more than two regression methods for TTP [Bin *et al.*, 2006; Sinn *et al.*, 2012], and there is simply one focusing on ensemble models [Mendes-Moreira *et al.*, 2012]. Table 2.3 presents a comparison between the aforementioned regression methods on this specific context. This comparison follows Section 10.7 in [Trevor *et al.*, 2001].

Recently, promising trajectory-based models employing ML techniques are being proposed to address TTP in this context. Reference [Tiesyte and Jensen, 2008] presents a Nearest-Neighbor Trajectory (i.e., based on a kNN model) technique that identifies the historical trajectory that is most similar to the current,

[3] It is a *black box-type* function that just provides an output and not a relationship between the independent variables and the target variable

[4] In this context, outliers may be trips with TT largely higher than expected due to some random event or other technical reason.

Table 2.3: A comparison between regression methods used to solve TTP-problems. Key: ⩔ - *poor*, ⩓ - *good*, • - *fair*. Based on Table 10.1 in [Trevor *et al.*, 2001].

Characteristic	**ANN**	**SVM**	**Trees based**	**kNN**	**kernel based**	**PPR**	**Linear Reg. Methods**
Parameter Sensibility (to variations on the parameters' set)	⩔	⩔	⩓	•	⩓	⩔	⩔
Handling of missing values	⩔	⩔	⩓	⩓	⩓	⩓	⩔
Robustness to outliers on the training data	⩔	⩔	⩓	⩓	⩓	⩓	•
Computational Scalability (handling large input datasets)	⩔	⩔	⩓	⩔	⩔	⩔	⩓
Ability to establish linear relationships between features	⩓	⩓	⩔	•	•	⩓	⩓
Ability to establish non-linear relationships between features	⩓	⩓	⩔	•	•	⩓	⩔
Interpretability	⩔	⩔	⩓	⩔	⩔	⩔	⩓
Predictive Power	⩓	⩓	⩔	•	•	⩓	⩔

partial trajectory of a vehicle. A TTP is provided by inferring the future trajectory of a vehicle. Similar trajectory-based approaches are proposed in [Lee *et al.*, 2012; Dong *et al.*, 2013]. However, such approaches are not applicable to long-term TTP because they mainly provide techniques that depend highly on the information available on the route segment already cruised by the vehicle (i.e., while the trip is in progress).

(B-b) State-Based and Time Series Models

These type of approaches just rely on the most recent data samples, disregarding the remaining historical data. The time series models assume that the TT is a linear/non-linear combination of its historical values [Cryer and Chan, 2008]. The state-based approaches usually assume that the future state of the dependent variables only relies on the most recent states. When compared to the other data-driven methods described previously, the present methods do not depend as much on the quantity of data and they do not require a large training period, since they mainly represent *Online Learning* algorithms (presented in detail in Chapter 3). Consequently, they are powerful short-term predictors due to their ability to *learn* and update in real-time, which does not occur with batch learning methods such as the ANN or kNN [Gama, 2010; Carrascal, 2012; Kieu *et al.*, 2012]. Nevertheless, the performance of these reactive models deteriorates when facing longer forecasting horizons (e.g.: [Park and Rilett, 1999]). An overview of the most commonly used state-based/time series models is presented in Table 2.4. Time Series models assume that the future TT on a given route depend only on its historical values [Jeong and Rilett, 2005]. The strength of these models is their high computational speed. However, they are commonly said to be unable to be built over online data, but only on historical data [Kieu *et al.*, 2012]. Despite being widely used for traffic flow prediction [Ihler *et al.*, 2006; Williams and Hoel, 2003; Min and Wynter, 2011], time series models are not so common in bus TTP. One of the explanations may be

Table 2.4: (B-b) State-Based and Time Series models commonly used to solve TTP problems.

Publication	**Denomination**	**Description**	**Examples**
[Markov, 1954]	Markovian Estimation Models	State-based estimation models where the conditional probability distribution of future system states depends solely on the present state.	[Lin and Bertini, 2004], [Sun *et al.*, 2007], [Rajbhandari, 2005]
[Kalman, 1960]	Kalman Filter	Recursive method to compute noisy data in order to determine the underlying system state.	[Cathey and Dailey, 2003], [Yu *et al.*, 2010], [Shalaby and Farhan, 2003], [Vanajakshi *et al.*, 2009]
[Box *et al.*, 1976]	AutoRegressive Integrated Moving Average (ARIMA)	It combines the most recent samples from the series to produce a forecast by following the historical data autocorrelation profiles.	[Rajbhandari, 2005], [Suwardo *et al.*, 2010]
[Holt, 2004]	Exponential Smoothing-based models	It proposes a way of calculating average historical samples in a time series by exponentially weighting each sample according to its age.	[Chen *et al.*, 2011]

their high sensitivity to changes in the relationship between historical and real-time data, especially when a stationary data distribution is assumed [Cryer and Chan, 2008]. Rajbhandari [2005] proposed an AutoRegressive (AR) model to capture the temporal variations of bus TT, while [Suwardo *et al.*, 2010] proposed Moving Averages (MA) in the same context. A self-adaptive exponential smoothing-based algorithm was proposed for interzone link TTP [Chen *et al.*, 2011].

State-based models are widely reported in TTP literature because they are capable of handling congested traffic situations [Carrascal, 2012]. The most commonly used state-based model is the Kalman filter [Wall, 1998; Cathey and Dailey, 2003; Shalaby and Farhan, 2003; Chen *et al.*, 2004; Vanajakshi *et al.*, 2009; Yu *et al.*, 2010]. Its main advantage comparatively to Markovian approaches is its ability to filter *noise* in the data, which is extremely relevant in Online Learning tasks for short-term prediction problems. Cathey and Dailey [2003] turns a sequence of AVL measurements into a sequence of vehicle state estimations (i.e., vehicle speed) to predict the arrival time by employing a Kalman filter. A similar approach was followed in [Vanajakshi *et al.*, 2009] and tested using data from buses running in Chennai, India. A model based on two Kalman filter algorithms was developed by [Shalaby and Farhan, 2003] to predict running and dwell times alternately in an integrated framework. Such filters use real-time AVL and APC data, respectively.

Lin and Bertini [2004] employed a simple Markov chain to predict trip arrival times at each bus stop by formulating a probabilistic transition model between "on schedule" / "behind schedule" states (which will represent the probability of the bus getting back on schedule during the remaining trip). A Finite State Machine was employed by [Sun *et al.*, 2007] based on the very same concepts. A Markov model to predict the propagation of bus delays to downstream stops is proposed by [Rajbhandari, 2005] for TTP.

In fact, these types of online models are not capable of dealing with long-term TTP alone. However, some works suggest that these models can be used as a complement to regression models [Wall, 1998; Chen *et al.*, 2004; Yu *et al.*, 2010; Zaki *et al.*, 2013]. Such approaches are promising. The regression models can handle complex relationships between multiple dependent variables by analyzing historical data. The Online Learning models are capable of using the stream of GPS data to refine these predictions. Commonly, this type of *hybrid* models employs the Kalman filter as an online building block. To the best of our knowledge, Wall [1998] were the first to suggest this in 1998. The authors employed a linear regression model to handle the TTP, while the Kalman filter was simply used to track the exact vehicle position based on the real-time stream of AVL data, which was not very accurate at the time. In [Chen *et al.*, 2004; Zaki *et al.*, 2013], the Kalman filter is proposed to fine-tune the TTP prediction produced by an ANN model based on APC data, while the work in [Yu *et al.*, 2010] does the same with an SVR model. Table 2.5 shows a high-level comparison of the AVL-based TTP models presented in this subsection.

(C) Setting Optimal Frequencies

In high frequency routes, the arrival times are not that relevant to the passengers' perception of quality service, and even for operational planning and control. Instead, optimal frequencies are set for such routes and the reliability studies on these routes usually try to find whether the headway is stable [Hounsell and McLeod, 1998]. However, research on setting optimal frequencies in bus

Table 2.5: High-level comparison of short-term TTP models.

Methods	**Advantages**	**Disadvantages**	**Reference Works**
ANN, SVR, PPR.	Good prediction results; Ability to discover non-linear relationships.	Low interpretability; High volumes of quality data are required.	[Chien *et al.*, 2002]
Trees-based Regression	Non-parametric; High scalability and Interpretability.	Low Predictive Power.	[Mendes-Moreira *et al.*, 2012]
kNN, Kernel-based Regression	Non-parametric; Handle missing data and Outliers.	Low interpretability; High volumes of data are required.	[Sinn *et al.*, 2012]
Trajectory-based kNN	Good approximation to the vehicle future's trajectory.	Do not handle long-term TTP.	[Tiesyte and Jensen, 2008]
Other statistical /Reg. methods	Simplicity; High Interpretability.	Low Predictive Power; Non-linear relations are not dealt with.	[Bertini and El-Geneidy, 2003b]
Exponential Smoothing	Simplicity.	Long-term TTP and non-linear relations are not dealt with. Low Predictive Power.	[Chen *et al.*, 2011]
ARIMA	High computational speed.	Long-term TTP is not dealt with. Highly Sensitive to Outliers.	[Suwardo *et al.*, 2010]
Markovian Models	Ability to handle *unknown* system states.	Long-term TTP is not dealt with.	[Lin and Bertini, 2004]
Kalman Filters	Ability to filter *noisy* data	Long-term TTP is not dealt with.	[Shalaby and Farhan, 2003]

timetables based on historical AVL data are scarce. Firstly, a formal definition of the problem is presented based on [Ceder, 2007; Hadas and Shnaiderman, 2012]. Then, research on this topic is presented.

Setting an optimal frequency is a compromise between passenger demand and resources available. Let L_0 be the desired occupancy of the vehicles operating on a given high frequency route of interest with n bus stops, i.e., $\{b_1, ..., b_n\}$ during a time interval T between two time instants (t_i, t_j). Then, let $\bar{d}_{b_k}$ be the average demand on a bus stop b_k during such period T and N be the number of departures available in the same period. The optimal headway in this interval, i.e., $H_{(t_i,t_j)}$, can be obtained as follows:

$$H_{(t_i,t_j)} = \min\left\{\max\left(\frac{L_0 \times T}{\max(\bar{d}_{b_k})}, \frac{T}{N}\right), H_0\right\} : k \in \{1, ..., n\} \tag{2.13}$$

where H_0 represents the *minimum* service level on such period. Obviously, research can be employed to determine the demand levels $\bar{d}_{b_k}$ based on AVL data.

Patnaik *et al.* [2006] presents a two-fold methodology to set optimal headways. Firstly, the APC data is clustered using Hierarchical Clustering. Each cluster corresponds to an optimal headway plan. Then, a Classification Tree is employed to discover rules to classify new instances (i.e., trips) into one of the available headway plans. A promising approach is introduced by Hadas and Shnaiderman [2012]: the optimal frequency setting model presented is based on the theory of supply chain models. The AVL data is used to model the statistical distributions of both demand and TT. Even though these works present useful insights on headway tuning, we do believe that there is room to explore AVL data in this specific context.

It is well known that even an optimal Schedule Planning cannot handle all the problems that arise while the network is operating, especially in high frequency routes. The next Section presents a summarized review of AVL-based methods to improve operational control in mass transit companies.

2.1.4 Automatic Strategies on Operational Control

The large-scale introduction of AVL systems in the bus fleets around the globe opened new horizons to operational controllers. This technology made it possible to create highly sophisticated control centers to monitor all the vehicles in real-time. However, this type of control often requires a large number of human resources, who make decisions on the best strategies for each case/trip. In the last decade, researchers started to explore the historical AVL data to build automatic control strategies, which can maintain the buses on schedule while reducing the human participation on the decisions.

This Section addresses the AVL-based automatic control strategies. Firstly, the four corrective actions typically recommended by the controllers to the vehicle operators are defined. Then, a systematic review of these automatic control strategies is both presented and discussed.

Corrective Actions

There are four typical methods employed as real-time control strategies [Strathman *et al.*, 2000]. These methods are typically (but not only) applied to highly frequent routes. They can be enumerated as follows:

1. **Bus Holding**: It consists of forcing the driver to increase/reduce the dwell time on a given bus stop along the route;
2. **Speed modification**: This strategy forces the driver to set a maximum cruise speed on its course (lower than usual on that specific route);
3. **Stop-Skipping**: Skip one or more route stops; also known as *short-cutting* when it requires a path change to reduce the original length of the route.
4. **Short-Turning**: This complex strategy consists of causing a vehicle to skip the remaining route stops (usually at its terminus) to fill a large service gap in another route (usually, the same route but in the opposite direction). In a worst case scenario, the passengers may be subjected to a transfer.

Bus-Holding control strategies are the most classic way of maintaining the buses on time. However, this headway alignment is made by increasing the TT of the passengers running in the vehicle [Strathman *et al.*, 2000]. The same applies to speed-based techniques. Moreover, stop-skipping/short-turning techniques align the service headways at the cost of the passengers who have to wait at the stops that were skipped [Liu *et al.*, 2013].

It is possible to divide the existing bus holding approaches into two main types [Fu and Yang, 2002]: (i) models that determine holding times on the basis of a mathematical control formulation with an explicit objective function, such as minimizing total passenger waiting time, and (ii) threshold-based control models where buses are held at a control stop on the basis of the deviation of the current TT from the scheduled headway. These models may assume theoretical values of dwell-time, TT or passenger demand (i.e., deterministic models) or assume that such events occur randomly (i.e., stochastic models).

A Review of Automatic Control Strategies

Eberlein *et al.* [1999] presents a study on three types of control strategies: holding, short-turning, and stop-skipping. The authors considered a one-way loop light-rail transit network of two terminals and n intermediate stations. In [Eberlein *et al.*, 2001], the optimal holding time at each stop is formulated as a deterministic mathematical optimization problem. The continuous characteristics of this problem were approached by employing a sliding window. This methodology was easily adapted from the originally studied light train to a bus network by [Zolfaghari *et al.*, 2004]. Hickman [2001] presented an analytical model for optimizing the holding time at a given control point in the context of a stochastic vehicle operations model. The author formulated the problem as a convex quadratic program in a single variable, and it is solved using gradient techniques. Zhao *et al.* [2001] present a multi-agent approach which was based on a negotiation between the bus and the stop agent to address the optimal holding problem.

Fu and Yang [2002] proposed a theoretical relationship to express the optimal holding time at a bus stop based on the current variation of the bus headways and the expected passenger waiting time. A similar approach was followed in

[Sun and Hickman, 2008] where the optimal holding problem is formulated according to deterministic variables such as the passenger arrival rate, the number of alighting passengers or the regular dwell time at a bus stop. Then, heuristics were proposed to solve this optimization model. The authors concluded that multiple holding locations are beneficial for minimizing total passenger-time, as opposed to the work in [Eberlein *et al.*, 2001], which assumed that only the original terminal should be used as control point. Again, it maintains the limitation of assuming deterministic variables.

The works in [Chen and Chen, 2009; Cats *et al.*, 2010; Li *et al.*, 2011b] also propose dynamic bus holding models to avoid headway irregularities in high frequency routes. The methodology was tested using AVL-based simulations which assumed stochastic distributions for the decision variables. Delgado *et al.* [2009] also suggested preventing passengers from boarding by establishing maximum holding times to maintain the headway stable. A regression model is proposed in [Yu and Yang, 2009] to deal with the holding problem: a SVR-based method forecasts the early bus departure times from the next stop based on four input variables (time-of-day, segment, the latest speed on the next segment, and the bus speed on the current segment). Then, an optimization model is employed to determine the holding time in each (bus, station) pair.

Only a small portion of data based works have employed other preventive actions than changing the bus holding time. Daganzo and Pilachowski [2011] proposed a multi-agent system where each bus would cooperate with the following to negotiate a maximum cruise speed to maintain the headway reliable. Sáez *et al.* [2012] proposed a dynamic discrete objective function that can detect disturbances in the headway regularity at each stop by employing a genetic algorithm. Stop-skipping and bus holding are suggested to the driver according to these events. The increase in the TT for the passengers on board caused by the holding strategy is taken into consideration in these last optimization models. Finally, Liu *et al.* [2013] proposed to study the short-turning as sub-problem of stop-skipping. A mathematical model is proposed using cost-based variables, such as the passenger waiting time or the TTV. Their main contribution is that they remove the common assumption of a *deterministic* bus TT, which is unrealistic due to the real-world influence introduced by road traffic conditions. Table 2.6 presents an overview of these automatic control frameworks.

By analyzing the existing literature on this topic, it is reasonable to conclude that this is still an open research field. Even if there are consistent studies on the holding problem, the four preventive actions were not regarded simultaneously in these works. Moreover, AVL data was used mainly as a proof of concept or to feed some statistical distributions on stochastic variables, such as TT or passenger demand. Another issue that is not broadly discussed in the literature is the threshold definition [Fu and Yang, 2002] (for instance, the minimum level of headway accepted, H_0, or the maximum service gap tolerated), to define when a control action should be adopted or not.

Despite the intrinsic chaotic characteristics of learning the headway instability problem, most techniques employed are mainly adapted to batch or online models which do not consider historical and real-time AVL data simultaneously. Even if such models are able to detect the concept drift often introduced by the unexpected events which occur in the system, such as traffic jams or a massive demand, few works have reported their deployment in a real-world bus network.

Table 2.6: Comparative Analysis on AVL-based Automatic Control Frameworks. Key: S - Stochastic, D - Deterministic, A - Agent-based, R - light-rail based.

Publication	Model Type	Evaluation Indicators			
		Bus Holding (1)	Speed Modification (2)	Stop Skipping (3)	Short Turning (4)
([Eberlein *et al.*, 1999])	D;R	√		√	√
([Eberlein *et al.*, 2001])	D	√			
([Hickman, 2001])	S	√			
([Zhao *et al.*, 2001])	A	√			
([Fu and Yang, 2002])	D	√			
([Zolfaghari *et al.*, 2004])	D	√			
([Sun and Hickman, 2008])	D	√			
([Delgado *et al.*, 2009])	S	√			
([Chen and Chen, 2009])	S	√			
([Yu and Yang, 2009])	S+*Reg.*	√			
([Cats *et al.*, 2010])	S	√			
([Li *et al.*, 2011b])	S	√			
([Daganzo and Pilachowski, 2011])	A		√		
([Sáez *et al.*, 2012])	S	√		√	
([Liu *et al.*, 2013])	S			√	√

2.2 Operational Control on Taxi Networks

Taxi service plays a vital role in public transportation by offering passengers a quick personalized destination service in a semi-private but secure manner. In Shanghai, for instance, taxi service takes up 23% of the total traffic volume while a company in New York reports an average driver income per shift of \$158 [Shanghai Municipal Statistical Bureau, 2008; Schaller Consulting, 2006]. Despite providing a convenient door-to-door service, taxi fleets are known by being highly inefficient (e.g. [Cheng and Nguyen, 2011] reports that 50% of its time is spent in idling state). The rapid growth of wireless sensors and development of Global Positioning System (GPS) technologies make it easier to obtain the timestamped spatial data reporting the vehicles' journeys. Typically, these taxis will report their locations in a certain but short frequency. Data such as its geo-position, timestamp, occupancy information is constantly recorded (using some weight-based sensor or by connecting the taxi meter to the communicational framework).

The Taxi industry have took this technological advances later than the mass transit companies (at the best of our knowledge, there is no relevant GPS-based work on improving the Operational Control on Taxi Networks before 2001) due to two main reasons: the 1) relation between the GPS price and its accuracy and 2) the personalized characteristics of this transportation service (which requires a lower level of coordination than the bus-based, for instance). Recently, the large increase on the fuel cost have particularly forced this industry to improve their operations and, more specifically, their control. This factor impelled both academic and industrial researchers to deeply analyze this spatiotemporal data to obtain ways to improve their profitability by A) reducing their vacant cruise time, B) improving their service routing profitability/reliability and C) dispatching. The C) Dispatching service was the addressed by improving the quality of the vehicle selection as well as the passenger waiting time.

The Automatic Vehicle Location and Dispatch Systems (AVLDS) (also known as Telematics) started by being systems which were able to automatically assign a service to the nearest available car (from the pick-up point demanded). One of the first reported systems to do so was located in Singapore (which has one of the largest taxi fleets in the world), where three companies adopted these AVLDS systems [Liao, 2001, 2003]. Then, customer-centered services started to be designed based both on the location and communication systems installed along such fleets. Booking in advance or automatic response systems improved the customers experience while ordering a taxi service. Such systems had a positive impact into reducing the time consumed by the dispatching tasks. In addition, these systems were able to collect data from each driver experience such as the GPS data, but also up-to-date information on the traffic conditions, accidents and jams, for instance (which can be used to improve routing models and/or functions) [Lee *et al.*, 2007]. The call's operators also benefit from this system by reducing misunderstandings and other entropies usually related to these kind of work. The number of attended calls is also increased, while the passenger load is fairly distributed along the fleet.

This Sections presents an overview on the GPS-based works focused on the Operational Control in Taxi Networks. The Section 2.2.1 is focused on works which evaluate the reliability, the performance and the underlying patterns of the service provided. Section 2.2.2 deeply revises the routing applications on reducing the vacant cruise time and/or improving their service routing profitability. Finally, Section 2.2.3 analyses the works focused on predicting the spatiotemporal distribution of the passenger demand.

2.2.1 Analysis of Service Performance

Taxi services may fall in one of three categories: 1) *dispatching*, where services directly demanded to a control center (e.g. by a phone call) are dispatched to a vehicle, which will pick-up someone in a nearby location; 2) *cruising*, where a taxi service is demanded directly, on the street and while the taxi is cruising (typically by a passenger waving to a taxi driver); 3) *standing* where the taxi is parked on a stand waiting for a passenger to get in. Nevertheless, the driver's success is highly based on its mobility intelligence. In large-scale cities such as New York or Beijing, the drivers are known to be *experts* on the demand spatiotemporal variability on a given city zone or area - and not in the entire city. However, they are not always located on their *comfort area* when they made their decisions (e.g. *Which road/stand should I head to pick up my next passenger?*). In fact, such decisions were only based on their own experience. The taxi's control centers (often defined as simple call centers) emerged to face this problem. Even so, they were not able to *mine* such large-scale mobility patterns. Instead, they provide a convenient but basic on-demand service - which is not enough to compete with other transportation modes nowadays (specially to perform daily connections). Such lack of information lead to an high inefficiency on managing the ratio between the available resources (i.e. vehicles and fuel) and the existing mobility needs.

Many past research efforts were employed to understand (and improve) the taxi fleets' efficiency. A comprehensive study presented by Yang and Wong

[1998] presented a framework to understand the equilibrium properties of taxis running in a network (and its relation with their efficiency). An unbalanced ratio between the passenger demand and the taxi's offer may lead to one of these two scenarios: I) an excess of vacant taxis or II) a larger waiting time to pick-up one of these. Nevertheless, the main reason for the service' inefficiency is the poor taxi driver's mobility intelligence. Such undesirable but typical behavior provokes a low ratio between the *live miles* (miles with fare) and *cruising miles* (miles without a fare). Some studies report that drivers' bad decisions may lead uo to 35-60% of the *cruising miles/total miles* ratio [Powell *et al.*, 2011], while others explain such variability on human factors such as the driver's age [Hong-Cheng *et al.*, 2010].

The historical archived location-based data of mobility traces can provide significant information, such as geographical distribution, time varying density of road traffic and passenger demand, link speed, destination estimation. It is also essential to implement many ITS applications by analyzing the underlying patterns on fuel wasting or urban mobility [Hoque *et al.*, 2012]. Although its numerous possibilities, the research on this topic is still very recent.

One of the first GPS-based works proposed to analyze and generate location histories is presented by Hariharan and Toyama [2004], where they classified their historical locations into *stays* (i.e. spots where a vehicle has spent some time) and *destinations* (i.e. clusters of stays). A first-order Markovian model was employed to do so. However, such model did not considered the specificities of the taxi trajectories (e.g. the concept of *staying* or *destination* may be irrelevant). Multiple spatial and temporal statistics of taxis' waiting spots were extracted from the historical data by Lee *et al.* [2008]. They also conducted a spatial clustering to identify some hot spots such as city hall, airport, central road and a shopping mart and an analysis on the waiting time on each stop as well as the success ratio. Cheng and Nguyen [2011] demonstrated a strong relationship between driver's movements and the relative attractiveness of neighboring regions. They did so by developing a multi-agent-based simulation framework which can be fed by real-world operational data. Such framework provide an opportunity to evaluate routing strategies (for the taxi drivers) as well as new polices and/or mechanisms. A driver-based analysis is provided by Liu *et al.* [2010a,b]: the drivers are divided between ordinary and top, based on their skills/income. A spatiotemporal pattern is mined from the top drivers' data where the primary focus is to reveal top driver mobility intelligence. The Traffic Ratio Density was computed to characterize the level of taxi services for each street district and facilitate the mapping of its spatiotemporal structure from data acquired from 9921 taxis running in Shanghai [Deng and Ji, 2011]. Another interesting but recent analysis of taxi mobility patterns is presented by Hoque *et al.* [2012], by monitoring multiple metrics such as instantaneous velocity profile, spatiotemporal taxi distribution, frequency distribution of pick-up and drop-off or hotspots identification.

The existing literature on this topic is mainly performed on an industrial perspective. However, very recent works have started to focus on the passenger opinion. Tung *et al.* [2011] presented a novel index to measure the comfort provided by the taxi service available on Taipei. One of the main problems on this industry is the fraud - it typically happens when a passenger is not a local and it consists into cruising a route between a given origin/destination highly larger than it could be. This way, some greedy drivers have the opportunity to overcharge tourists. A first approach to this problem was presented by Chen *et al.* [2012], where an anomalous route detecting model is proposed. Ge *et al.* [2011] presented an innovative study on this topic where two fraud evidence measures were proposed: a travel route evidence and a driving distance evidence. Statistical models were thereby developed to introduce an algorithm to generate a typical driving path from one interesting site to another.

Such analysis highlight the importance to improve the taxi driver's mobility intelligence. Methods to do so by providing intelligent routing and other GPS-based recommendation systems are presented in the next Section.

2.2.2 Intelligent Routing and Recommendation Systems

The Intelligent Routing problem concerns the definition of a route through a given origin/destination which could pass by some Points of Interest (POI) and/or into finding the fastest (or the shortest) path between such origin/destination. However, in this Section we review works focused on where the passenger demand will rapidly emerge. This can be computed using one of two approaches: 1) as a routing problem, where the defined route has an high likelihood to provide service demands; 2) as a recommendation model, where the target variable is a zone or a stand where the demand will *certainly* be high. Many of the routing works and/or frameworks reported on literature rely on the Dijkstra's work and/or on the A^* algorithm. The recommendation problem - as well as its relation with the routing one - is formally enunciated below.

Let $S = \{s_1, s_2, ..., s_N\}$ be the set of N taxi stands of interest and $D = \{d_1, d_2, ..., d_j\}$ be a set of j possible passenger destinations. The 2) recommendation problem consists into choosing the best taxi stand at instant t according to our current *sensing* (independently on how it is obtained) about passenger demand distribution over the time stands and/or urban regions for the period $[t, t+P]$, as illustrated in Fig. 2.4. The 1) routing problem extends such definition by adding the notion of *POI - Point of Interest.* Consequently, the problem is not only to decide which should be the stand $s_l : l \in \{1, ..., N\}$ to head to after a passenger drop-off in a destination $d_k : k \in \{1, ..., j\}$ but also to find a set of z POIs $\mathcal{I} = \{I_1, ..., I_z\}$ along the route between d_k and s_l where taxi-passenger demand can rapidly emerge.

Typically, the models based in either one of these approaches aim to maximize the likelihood of picking-up the next passenger as soon as possible. However, some works assume that the taxis cruise *randomly* to find their next passenger - therefore, the models provide a route that aims to reduce the vacant *cruising miles* as much as possible [Powell *et al.*, 2011]. An illustration of a real world instance of the problem is displayed in the Fig. 2.5. After a drop-off, the driver needs to choose one of the nearby stands (represented by blue dots).

The taxi historical trajectories contain two main types of knowledge [Yuan *et al.*, 2011b]: the a) passengers' mobility (i.e. where and when the passengers were picked-up/dropped-off by a taxi) and b) the taxi drivers' pick-up behaviors (and its mobility intelligence). Yue *et al.* [2009] proposed to mine the Level of Attractiveness (LoA) of each region/zone by defining a time-dependent origin/destination matrix. They demonstrate its usefulness by quantizing the attractiveness among clusters. Such results could facilitate our understanding about the mobility in a city and they are commonly used as input to this routing/recommendation models. Chang *et al.* [2010b] presented a novel insight on demand prediction: they applied clustering to data extracted from large Asian cities. They used some key features besides location/time such as the weather. Their output was a hotness probability ratio over spatial clusters (i.e. real agglomeration of roads/streets) dependent on the driver location, discarding however the other taxis position.

Li *et al.* [2009] proposed a hierarchical GPS-based routing method which is able to *rank* the road segments based on the frequency of their use (i.e. an highly used road will be a road where many taxi drivers pass by). Secondly, a graph-based algorithm determines the *best* route based on the drivers' experience. An innovative study was presented by Li *et al.* [2011a]. Their goal was to validate the triplet Time-Location-Strategy as the key features to build a good passenger finding strategy. They used a L1-Norm-SVM as a feature selection tool to discover both efficient and inefficient passenger finding strategies in a large city in China. They made an empirical study on the impact of the selected features and its conclusions were validated by the feature selection tool. Lee *et al.* [2008] constructed a recommendation model based framework to describe the spatiotemporal structure of the passenger demand on Jeju Island, South Korea. Zhang *et al.* [2012] presented a spatiotemporal clustering framework able to both mine and select the top-5 valuable pick-up points/clusters. A time-dependent landmark graph is proposed by Yuan *et al.* [2010] (a landmark is considered as a road segment frequently traversed by taxis). Such graph is used to model the taxi drivers' mobility intelligence and the links' weights are calculated by a clustering process which is able to estimate the distribution of travel time between two landmarks in different time slots. A Cloud-based system computing was proposed by Yuan *et al.* [2011a]: this model is able to aggregate and mine information from the taxi' network but also from other sources on the Internet such as Web maps and Weather forecasts. Based on such data, the model is able to predict the short-term traffic conditions and to provide the fastest route to perform the service. This framework was evaluated using data from 33000 taxis running on the city of Beijing, China.

Powell *et al.* [2011] proposed to reduce the drivers *cruising time* by providing a Spatiotemporal Profitability map which is able to suggest the most profitable

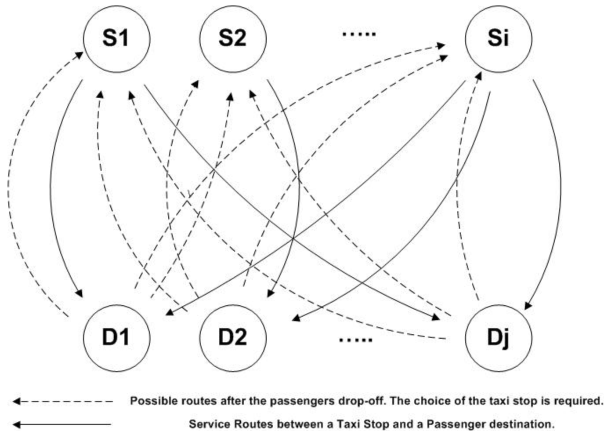


Figure 2.4: Intelligent Routing over taxi networks: a problem illustration.

Figure 2.5: Illustration about the Recommendation of an highly profitable stand.

regions on the map based on historical data, the drivers' current position and the present time. Ge *et al.* [2010] provided a cost-efficient route recommendation model which was able to recommend sequences of pick-up locations. Their goal was to learn from the historical data transmitted from the most successful drivers to improve the profit of the remaining ones. Yuan *et al.* [2011b] presented a very complete work containing methods about a) how to divide the urban area into pick-up zones using spatial clustering; b) how a passenger can find a taxi; and c) which trajectory is the best to pick-up the next passenger. Although their results are promising, all approaches are focused on improving the trajectory of a single driver, discarding the current network status (i.e. the position of the remaining drivers) as well as the real cost (e.g. fuel) to get to the pick-up point.

Hu *et al.* [2012] proposes an innovative cost-saving model: it firstly mines interesting time-dependent pick-up points by clustering the historical GPS traces; secondly, a skyline computing-based heuristic computes a pick-up tree where the taxi current location is the node able to connect all the interesting pick-up points (i.e. centroids). Thirdly, a probability model to estimate the *fuel consumption* is presented and employed as the weight of every route. However, the position of the remaining cars is also discarded by this recommendation framework. Typically, this recommendation depends on four main variables:

1. the apriori distance from our current location to the area/stand we could head to (it can also be expressed as a cost in currency and/or time);
2. the *expected* service revenue that we will pick-up on that specific location;
3. the passenger demand *expected* to emerge on such area/stand in the next period of $P - minutes$;
4. how many vacant vehicles are already parked and/or cruising on such stand/area.

At the best of our knowledge, there is no work in the literature proposing a recommendation model able to handle these four variables in real-time. While it is reasonable admit that the variable 4 can be measured in real-time, the remaining ones need to mined from the network's historical GPS data. In fact, a more realistic estimation of the distance could be to *estimate* the cost to cruise such path to this specific area (like it is proposed in [Hu *et al.*, 2012]). A similar approach could be performed to the variable 4, taking full advantage about the underlying patterns contained in the previously acquired data. Recently, a few innovative works regarding the prediction problem on the short-term passenger demand at a given area/stand (as well as the number of cruising vacant vehicles) were proposed. Such works can be faced as a straightforward contribution to the existing recommendation models. They are briefly reviewed in the following Section.

2.2.3 On Predicting Service Spatiotemporal Patterns

The knowledge of the number of vacant/occupied taxis in different areas in the city as well as its short-term state provides the information for a better scheduling. For example, a tourist who arrives at an airport in a transit city and wants to make a trip inside the city with limited time will benefit from the service by using it to plan out a series of taxi rides around the city. We will briefly revise this kind of predictive models over the spatiotemporal distribution of both the vacant taxis and the passenger demand throughout the city.

A promising framework to predict the number of vacant taxis was presented by Phithakkitnukoon *et al.* [2010]. They employed a classifier based on the day of the week, on a given time of the day and on the weather condition corrected by an Error-Based Learning method. A dataset containing 4 months of data from Lisbon, Portugal was successfully used as test bed. A similar work was presented by Mayuri and Rajesh [2013], where a probability function expresses the likelihood to find a vacant taxi on a given area based on historical data. However, this work also provide a probability model to estimate the inverse event (i.e. the likelihood to pick-up a passenger). In fact, the prediction of the passenger demand's spatiotemporal distribution is a hot topic.

Li *et al.* [2012] present a recommendation system to improve the driver mobility intelligence. To do so, they used data from a taxi network running in Hangzhou, China. Firstly, they calculated the city hotspots: urban areas where pick-ups occur more frequently. Secondly, they used ARIMA to forecast the pick-up quantity at these hotspots over periods of 60 minutes. Thirdly, they presented an improved ARIMA dependent both on time and daytype. Finally, they proposed a recommendation system based on the following variables: 1) the number of taxis already located at each hotspot; 2) the distance from the driver location to the hotspot in time and 3) the prediction about the number of services to be demanded in each one of them. However, their approach also presents some strong limitations: 1) it just uses the most immediate historical data, discarding the mid and long-term memory of the system; 2) their test-bed uses minimum aggregation periods of 60 minutes over offline historical data (i.e. the next value prediction task on a time series goes easier as long as you increase its aggregation period). Such method represents the historical time series as an histogram one. However, once we are in a bin, we will only have a novel prediction after passing to the next one (i.e. a recommendation model has to operate in real-time. Therefore, it is inadequate to consider that a prediction at 2pm, for instance, will be enough to decide at 2:05pm, 2:15pm or 2:35pm); 3) the paper does not clearly describes how they update both the ARIMA model and the weights used by it. Notwithstanding the validity of employing time series analysis techniques to handle this problem, further research to mitigate the abovementioned issues is needed to improve their applicability.

2.3 Challenges and Research Opportunities

Throughout this Chapter, the most significant contributions in AVL-based research to improve the service reliability on public transportation networks were presented. This Section discusses twelve **Research Opportunities** (i.e. **(a-l)**) to be explored in the future, as well as open challenges to the research community on these topics.

2.3.1 Improving Operational Planning on Mass Transit Agencies

Section 2.1.1 briefly revises the steps of the traditional OP. Even though AVL-based research has emerged recently on improving route definition, most AVL-based works on OP focus on the SP. The State-of-the-Art relies on deterministic and cost-based models. The AVL data makes it possible to perform a bottom-up OP evaluation, namely correctly exploring the available resources or even reducing them if possible to meet the current demand. A **(a)** complete AVL-based framework to re-design all the steps of the OP (already discussed in Section 2.1.1) is a research goal on this topic for the medium term future.

2.3.2 Evaluating SP reliability on Mass Transit Agencies

It is possible to identify two main issues where further AVL-based research should be employed to improve the evaluation of SP reliability: **(b)** creating an unique evaluation indicator, considering the company's perspective on the evaluation by including external factors in the evaluations, or by developing cost-related evaluations and to **(c)** evaluate the reliability of the current schedule's number and coverage. These subjects are described below.

The aforementioned evaluation metrics are classical but widely used in evaluation studies. However, distinct metrics (which are highly correlated to the main ones) are continuously emerging. It is known that the importance of each one of these indicators *depends* on the frequency established in the route. However, to the best of our knowledge, **(b)** there is no consensual, individual and integrated reliability ratio.

The first step in building a SP is defining both the schedule's number and day coverage. Then, a timetable is assigned to each schedule in a stepwise process already discussed in Section 2.1.2. This definition has an explicit impact on the definition of timetables. However, to the best of our knowledge, no research addresses the evaluation of whether the schedule's number and coverage still suit the current demand patterns and network behavior. Consequently, a question arises: **(c)** *Is it possible to assess whether the schedule's number and coverage is suitable for the network needs based on historical AVL data?*

2.3.3 Improving SP Timetabling on Mass Transit Agencies

In terms of improving SP timetabling based on AVL data, four subjects emerged as research opportunities as a result of the extensive review of the existing literature: fine-tuning the Schedule using **(d) long-term TTP** and **(e) optimal slack times**; **(f)** building **automatic** methods to perform **feature selection** for TTP problems, and **(g)** performing **before-and-after SP evaluations**. These opportunities are now described in detail.

A large gap identified in the literature has to do with the **(d)** AVL-based long-term TTP. The regression models represent the most relevant slice of the State-of-the-Art on AVL-based short-term TTP. However, some works have also demonstrated their usefulness in long-term problems [Mendes-Moreira *et al.*, 2012]. The AVL data makes it possible to explore these models to improve the SP. Such approaches can present a breakthrough for this research area over the next decade.

In Section 2.1.3, **(e)** the slack time is introduced in the SP to handle variability in TT. Prior to deploying Automated Data Collection systems in mass transit companies, computing that variability was a difficult task. However, the availability and the reliability of the historical AVL data used today represent a clear opportunity to improve the schedules using this well-known strategy - already being explored by a few set of recent works [Mazloumi *et al.*, 2012; Yan *et al.*, 2012; Jorge *et al.*, 2012].

Even though regression models are simple to apply, they suffer from several limitations in the context of TTP. The greatest limitation is that many variables in transportation are highly correlated [Jeong and Rilett, 2005; Wong, 2011]. However, there is not much research on the **(f)** automatic feature selection for TTP regression problems [Patnaik *et al.*, 2004; Mendes-Moreira, 2008; Mazloumi *et al.*, 2011]. This step can be particularly important to facilitate the training stage of complex regression models such as ANN or SVR.

Evaluating the changes performed on the SP is difficult prior to deployment. Even though this review discusses various works focused on improving the SP, not many of them evaluate the impact of the suggested changes. The **(g)** *before-and-after* evaluation studies are crucial to **quantify** the relevance of these adjustments. To the best of our knowledge, there is only one AVL-based study of this type [Tétreault and El-Geneidy, 2010; El-Geneidy and Surprenant-Legault, 2010]: [Tétreault and El-Geneidy, 2010] select bus stops and estimate run times for new express services, while [El-Geneidy and Surprenant-Legault, 2010] evaluates the reliability of the route SP after deployment.

2.3.4 Automatic Control Strategies for Mass Transit Agencies

To predict instability and unreliability in the network while the buses are operating is a difficult challenge. Not much research focuses on more than one preventive action. Moreover, the test-beds employed are mainly proof of concepts because they use limited data collections both in space (i.e., number of routes) and time. Yet, many mass transit companies have large collections of data in their databases whose potential is far for being fulfilled. This family of problems is closely related to the Online Learning problems. Even though they are common in TTP problems, ML techniques have rarely been applied to build automatic control strategies. Problem formulations, such as Stream Event Detection can represent a breakthrough in the Control area. Moreover if it is possible to combine Online Learning with patterns mined from historical data. Such hybrid methodologies (i.e. regression models plus state-based learning) are being proposed for short-term TTP in the context of Advanced Traveler Information Systems. However, **(h)** we believe that these two areas could be an interesting topic to explore in the Control context.

Recently, researchers focused on building efficient control systems capable of monitoring headway regularity and of avoiding inconvenient events, such as BB. Surprisingly, there is not much research on this specific topic. Many of the existing works focus primarily on the optimal bus holding problem, thus disregarding the remaining corrective actions. Consequently, one challenge arises: **(i)** *Is it possible to build a methodology that considers and selects one of the four known corrective control actions based on AVL/APC data?* In fact, such methodology addresses two distinct problems that are not conveniently covered in the literature: **(i-1)** *Is it possible to define an optimal control threshold?*[5]. **(i-2)** *How is it possible to choose the best corrective control action after the optimal control threshold is reached?*

2.3.5 Informed Driving Methods for Taxi Services

The taxi networks are more exposed to the stochastic phenomenons that affect traffic conditions and the service demand than the inner-city mass transit services. Consequently, their control tasks - such as service assignment or route selection - are highly relevant to maintain their service costs on sustainable levels. However, the introduction of advanced taxi dispatch centers on major urban areas is even more recent than on mass transit agencies [Lee *et al.*, 2007; Furth *et al.*, 2003]. Obviously, the most important variable to handle such issue is the short-term passenger demand. The State-of-the-Art on this topic is still very limited. Consequently, it is urgent to **(j)** develop methodologies able to accurately predict the future values of such variable able to handle such stochastic events.

Moreover, it is known that the driver's decisions are not only demand-based: it also concerns the current position of its competitors (i.e. other drivers) and the **(k)** apriori travel time estimation (i.e. before the trip starts) between each origin and destination (e.g. how long a given driver will take to pick-up a passenger on a specific urban area at a given time of the day). Therefore, a **(l)** recommendation model able to provide accurate information on each driver decision (namely, on finding their next passenger) in real-time can be highly impactful on this industry. However, at the best of our knowledge, there is no such model in the State-of-the-Art of this topic.

2.4 Overview and Research Goals

In this Chapter, the location-based ITS applications on improving both the planning and the control of mass transit transportation networks were revised. In

[5] When should a control action be taken to restore real-time service reliability (and avoid BB occurrences)?

the last decade, many relevant contributions were presented on this topic. The spatiotemporal features of this type of data provided new and unprecedented opportunities to reveal underlying patterns on unexpected behaviors and/or events which are deteriorating the schedule - and therefore, the service's quality. This data availability is now inexpensive and widely spread as a standard in every mid/large-sized inner city public transportation networks.

The necessity of having a good planning/control of public transportation networks increases along with the number of vehicles running on the largest urban areas. These companies face today a strong competition not only from the individual transport but also from the alternative modes such as the metro, the tram and the trains - specially considering the rising fuel costs at an worldwide scale. Such systems are now the truly competitive advantage which can draw a thin line between the financial success and the bankruptcy. More than a good operational plan, these networks have an emergent need to be adequately controlled (preferentially, without human intervention - or, at least, to reduce it as much as possible).

This need encouraged more and more researchers to focus on mining the GPS data broadcasted by the vehicles. This data source dramatically changed the way to improve both the operational planning and control on these networks. The theoretical models were progressively replaced by complex but efficient statistical models and ML algorithms. Nowadays, it is even more important to provide real-time information to the passengers about what is *happening* in the network (i.e. on-spot arrival time information). More than building an exact but time-consuming prediction on the arrival time, the researchers have focused into building simple frameworks able to constantly learn from location-based data streams. They did so by proposing online learning models (e.g. Kernel filters and Markovian models).

However, it was the operational control which benefited the most of these introduction. The old radios and communicational frameworks were now replaced by high tech large-scale monitoring centers where it is possible to observe the vehicles/drivers' behavior in real-time. Recently, the researchers focused their attention on building automatic but efficient control systems able to monitor the headway regularity and to avoid inconvenient events such as BB. Surprisingly, there are not so many works regarding the occurrence of BB. Many of the existing works on improving the operations' control are focused on the holding problem (i.e. to discover the optimal holding time of each vehicle/bus stop pair). Consequently, these works present a **reactive** nature by introducing corrective action just when the headway is already highly unstable (i.e. BB event). Many other related problems take similar approaches (e.g. offline taxi-passenger demand prediction, which does not handle bursty peaks). By generally anticipating these events using ML frameworks, we intend to adopt **proactive** approaches from which we take actions (e.g. stop skipping or vehicle re-routing) to avoid them instead of mitigating their effects.

This PhD thesis take an *explorative* approach to improving the planning and control of PT networks using GPS data. Firstly, we departed from a very general hypothesis (defined in Section 1.2) to review the existing State-of-the-Art of data driven methods on these topics. The aim of this step was to identify research opportunities in the literature where advances can be provided based on the location data broadcast by each vehicle. In the previous Section, twelve specific research topics were drawn from such review. Hereby, six of these topics are addressed. To do so, the following research goals were devised:

1. Explore unsupervised offline learning methods to reduce the travel time variability by automatically evaluating the schedule day coverage and identify improvement measures **(c)**;
2. Attempt to build online methodologies capable of **predicting Bus Bunching** events in real-time **(h)**;
3. To develop a framework able to automatically **select and deploy a corrective action** given a Bus Bunching alarm **(i)**;
4. Seek a real-time Time Series Analysis method able to **predict the short-term taxi-passenger demand behavior** over an urban area **(j)**;
5. Research online learning techniques capable to produce **real-time smart recommendations** about the most adequate to head to based on the current network status **(k,l)**;

Obviously, such goals imply the exploit of distinct data sources (i.e. taxis and buses networks). However, we believe that the data from such sources share some characteristics that justify such parallel study - namely, by developing methods able to use both sources simultaneously. Consequently, a secondary goal is also established as follows:

6. To **explore the synergies between the taxis and buses networks** by developing methodologies to meet any of the abovementioned goals able to **learn from both data sources**.

The decision support systems accomplished from the primary goals must be able to cope with the streaming spatiotemporal information. This aspect requires streaming data mining algorithms able to continuously maintain decision models consistent with the current state, monitor events in real-time, detect changes, etc.. We believe that the introduction of these real-time decision framework can be a novel and major contribution for public transportation networks.

In the next Chapter, we present a brief overview on adequate online methods for the problems described above.

Chapter 3

Fundamental Concepts on Learning from Data Streams

In the last decades, Machine Learning (ML) research have focused on *batch learning* (i.e. offline learning) usually using relatively small datasets. On *batch learning*, the training data is assumed to be entirely available to the algorithm - which outputs a decision model after processing the data multiple times. However, most applications require learning algorithms able to act *while* the data is being collected. Such algorithms have to be able to incorporate new data as it arrives - in an *incremental* manner. Moreover, most of these processes are **non-stationary** (i.e. their concepts evolve over time; e.g. a spam filter or an antivirus scanner) - therefore, it may be not enough to be incremental. A successful learning algorithm must be also able to handle **concept drift**, forget outdated data and adapt to the current state of **nature** [Gama, 2010].

A greater challenge is now upon the ML community by the introduction of automatic data feeds. Unlike the human-generated ones, these transient data streams have a particular but important constraint: it is not feasible to load the arriving data into a traditional database management system, which is not designed to directly support the **continuous queries** required by such applications. The traditional learning methods (i.e. offline) made some assumptions which are not compatible with these data streams such as *finite data sets*, *static models* and/or *stationary distributions.* These aspects are derived from novel aspects about this kind of data:

- The data is produced/broadcasted through *unlimited streams* that continuously flow, eventually at high speed, over time;
- The data distribution may be non-stationary (i.e. the underlying regularities may *evolve* over time);
- The data are now spatially situated (as well as time situated);

But could small adaptations to the traditional ML algorithms suffice to handle such new data characteristics? Even very basic operations (common to many of the most successful and widely used algorithms) are challenged with such new settings. For instance, we can consider a standard procedure to cluster variables. Typically, these variables are represented by columns in a working matrix. Such matrix can be clustered by applying any clustering algorithm over its *transpose*. However, in a scenario where the data evolve over time, we cannot use such trick (i.e. the transpose operator cannot be used [Barbará, 2002]). Therefore, the learner can only afford one pass on each data piece because of time and memory constraints. When the learner has to decide *on the fly* what is relevant and must be processed and what is redundant and could be *forgotten*?

Formally, we can define **Adaptive Learning Algorithms** as follows. Let $E_t = \{\vec{x_i}, y_i : y = f(\vec{x})\}$ be a set of examples available at time $\{1, 2, 3, ..., i\}$. A learning algorithm is adaptive if from the sequence of examples $\{..., E_{j-1}, E_j, ...\}$, produce a sequence of Hypothesis $\{..., H_{j-1}, H_j, ...\}$, where each hypothesis H_i only depends on previous hypothesis H_{i-1} and the example E_i.

An adaptive learning algorithm requires two operators:

- *Increment*: the example E_k is incorporated in the decision model;
- *Decrement*: the example E_k is forgotten from the decision model.

In summary, knowledge discovery from data streams implies the following requirements:

- The algorithms will have to use limited resources, in terms of computational power, memory, communication, and processing time;
- The algorithms may have to communicate with other agents on *limited bandwidth* resources;
- In a community of smart devices geared to ease the life of users in real time, answers will have to be ready in an *anytime protocol*;
- Data gathering and data processing might be *distributed*.

In this Chapter, we review some of the most well known techniques to learn from data streams useful for the problems approached on this thesis. This review is based on a comprehensive survey on this subject presented by Gama [2010]. However, some prior knowledge on ML basics is recommended - but not mandatory - to fully acknowledge its insights. To ease the interpretation of this Section, some definitions about computational learning methods are presented below.

- **Supervised Learning:** to infer a function from labeled training data (e.g. the price of a given product or a military rank). The training data usually consists into a set of instances with an input object (typically a vector) and a desired output value. Such function is then used to compute the value of novel examples where the output value is *unknown* [Mohri *et al.*, 2012];

- **Unsupervised Learning:** to find one (or multiple) hidden structure in unlabeled data. One of the most well known approaches to unsupervised learning is clustering [Mohri *et al.*, 2012];

- **Offline Learning:** a method able to learn a predictive model from a ***finite*** set of instances where the post-training queries do not improve its previous training [Burke *et al.*, 2010];

- **Incremental Learning:** a method able to learn and update its predictive model ***as long as*** the true labels of the input samples are known (i.e. a stepwise method where each step uses one or more samples) [Chalup, 2002];

- **Online Learning:** an *incremental* learning method which is able to update the model ***every time*** a true label of a newly arrived sample is known (i.e. it learns from one instance at time)[Burke *et al.*, 2010];

- **Real-Time Learning:** an online process able to operate in *real-time* (i.e. to use the last sample *true label* to update the predictive model ***before*** the next sample arrives)[Huang *et al.*, 2006];

This Chapter is structured as follows: the Section 3.1 presents some introductory methods and concepts to analyse a data stream. The third Section discusses some methods to maintain histograms on this context. The traditional Time Series Analysis techniques are presented in Section 3.3. Section 3.4 presents State-of-the-Art techniques to ensemble prediction methods working over a stream of data. Finally, we propose some evaluation metrics to work over models learned from data streams in Section 3.5.

3.1 Basic Streaming Methods

Data streams are unbounded in length and depth (i.e. the domain of possible values of an attribute can be very large). For instance, the domain of all pairs of IP addresses on the Internet: it is almost impossible to store all data and execute queries over this past data. Most of these types of queries require techniques to somehow summarize information about the past data. Such techniques usually require *O(N)* space...how can we use those in a *restricted* memory' (i.e. less than *O(N)*) scenario?

There are three main constraints to consider when we are querying data streams: 1) The amount of memory used to store the information; 2) the time to process each data element and 3) the time to answer the query of interest. A summary of the differences between traditional and stream data processing is presented in Table 3.1.

Algorithms that process data streams are typically *sub-linear* in time and space. However, its answer is in some sense *approximate*. In general, we can identify two types of approximate answers: 1) ϵ Approximation: the answer is correct within some small fraction ϵ of error; 2) (ϵ, δ) Approximation: the answer is within $1 \pm \epsilon$ of the correct result, with probability 1-δ. The constants ϵ and δ are strongly correlated with the space complexity of our solution. Typically, the space is $O(\frac{1}{\epsilon^2}\log(1/\delta))$.

In this Section, we will briefly present some basic techniques to handle learning from data streams on three distinct perspectives: 1) Poisson processes, 2) Sliding Windows and 3) Data Sampling and Summarization.

3.1.1 Poisson Processes

A typical example of a data stream is a Poisson process. We can define it as a stochastic process in which events occur *continuously* and *independently* from each other. We can easily observe real life examples of this such as the passenger hopping on/off buses on a given stop, telephone calls arriving or the number of meals delivered by a take-away restaurant.

A random variable x is said to be a Poisson random variable with parameter λ if x takes values 0,1,2,...,∞ with:

$$pl = P(x = k) = e^{-\lambda}\frac{\lambda^k}{k!} \tag{3.1}$$

$P(x = k)$ increases with k from 0 till $k \leq \lambda$ and falls off beyond λ. The mean and variance are $E(X) = Var(X) = \lambda$. The Poisson processes present some interesting properties such as:

- The number of points t_i in an interval (t_1, t_2) of length $t = t_2 - t_1$ is a Poisson random variable with parameter λ_t;
- If the intervals (t_1, t_2) and (t_3, t_4) are non-overlapping, then the number of points in these intervals are independent;
- If $x_1(t)$ and $x_2(t)$ represent two independent Poisson processes with parameters $\lambda_1 t$ and $\lambda_2 t$, their sum $x_1(t)+x_2(t)$ is also a Poisson process with parameter $(\lambda_1 + \lambda_2)t$. However, in many problems, we are not interested in maintaining statistics over all the past but only over the most *recent* one. One of the most used techniques to consider what is *old* enough to be *forgettable* are the sliding windows, described in the following Section.

3.1.2 Sliding Windows

When we want to consider just the most recent observations, a *sliding window* of fixed size is the most simple solution. Whenever an element j_i is observed and inserted into the window, another element j_{i-w} (where w represents the

Table 3.1: Differences between traditional and stream data query processing.

	Traditional	**Stream**
Number of Passes	Multiple	Single
Processing Time	Unlimited	Restricted
Memory Usage	Unlimited	Restricted
Type of Result	Accurate	Approximate
Distributed?	No	Yes

window size), is forgotten. Babcock *et al.* [2002] defines two basic types of sliding windows:

- **Sequence based**. The size of the window is defined in terms of the number of observations. Two different models are *sliding windows* of size j and *landmark windows*;

- **Timestamp based**. The size of the window is defined in terms of *duration.* A timestamp window of size t consists of all elements whose timestamp is within a time interval t of the current time period.

Note that to calculate statistics over sliding windows, we need to maintain all elements within the window in memory. Suppose a problem of interest where we want to maintain a window of 100 samples from 1000 observations $(x_1, x_2, ..., x_900, x_901, ..., x_1000)$. Consider the following sufficient statistics (i.e. statistics over the observations inside the window) after receiving the 1000th observation:

$$A = \sum_{i=901}^{1000} x_i; B = \sum_{i=901}^{1000} x_i^2 \tag{3.2}$$

Whenever the 1001th value is observed, the sufficient statistics will be updated as $A = A + x_{1001} - x_{901}$ and $B = B + x_{1001}^2 - x_{901}^2$. Even being easy to update this type of statistics we still need to maintain all the observations within the window in memory (independently of the window' size). However, this forgetting mechanism may not be enough to understand the current data' nature - specially if it carries seasonality somehow. Another interesting family of techniques to handle this need to maintain sufficient information about the past without overloading the memory is the **data reduction** one. We summarily describe some of these techniques in the next Section.

3.1.3 Data Sampling and Summarization

The Data Reduction techniques consist in mechanisms to define, maintain and update data structures which contain sufficient statistics about the past data. Two of the most commonly used techniques are the 1)*sampling* and 2)*histograms.* They are described below.

Sampling

Sampling is a common practice for selecting a subset of data to be analyzed. Actually, the Sliding Window techniques can be faced as a particular case of Sampling - where the only criteria to select if we want to maintain a given observation in memory is *if it is recent enough.* However, as the Sliding Windows, the Sampling processes also present strong drawbacks: while they reduce the amount of data to process, and, by consequence, the computational costs, they can also increase the number of errors. Therefore, the main problem is to define the criteria to obtain a *representative* sample - a subset of data that has approximately the same properties of the original data. Some of the most well-known techniques to do so are the *Reservoir Sampling* [Vitter, 1985], the *Min-Wise Sampling* [Broder *et al.*, 2000] and the *Load Shedding* [Tatbul *et al.*, 2003]. The first one is briefly described below.

The *Reservoir Sampling* takes one parameter, i.e. z, which defines the number of samples to be maintained from the original set of N samples, i.e. S. Then, a second set is created containing the first z samples in S, i.e. $R \subset S$. Thirdly, the remaining elements of S are also scanned. For each i_{th} value of S, a random number is generated, i.e. $r \in [1, z]$. If $r \leq z$, the r_{th} is replaced by the i_{th} value of S, i.e. $R[r] \leftarrow S[i]$. This incremental procedure gives the similar probability to every elements S to be included on R.

Histograms

Histograms is a summarization technique that can be used to approximate the frequency distribution of element values in a data stream. It is visualized as a bar graph that shows frequency data. Using a simplistic approach, we can build an histogram by sorting the values of a random variable of interest and placing them into bins (i.e a set of non-overlapping intervals). Each interval is defined by the boundaries and a frequency count (equal to the number of data points inside each bin). However, it can be difficult to maintain equally-sized boundaries from a data stream by two main reasons: firstly, it may be not possible and/or necessary to keep the same amount of information (i.e. equally sized bins by width/frequency) from the recent past rather than the oldest one. Secondly, we can be handling a random variable which do not follow a stationary distribution. In the following Section, we briefly discuss some specific techniques to maintain histograms from data streams.

3.2 Maintaining Histograms from Data Streams

Formally, we define a histogram as a set of break points $b_1, ..., b_{k-1}$ and a set of frequency counts $f_1, ..., f_{k-1}, f_k$ that define k intervals in the range of the random variable: $[-\infty, b_1],]b_1, b_2], ...,]b_{k-2}, b_{k-1}],]b_{k-1}, \infty]$. The most commonly used histograms are either *equal width*, where the range of observed values is divided into k intervals fo equal length $(\forall i, j : (b_i - b_{i-1}) = (b_j - b_{j-1})$, or *equal frequency*, where the range of observed values is divided into k bins such that the counts in all bins are equal $(\forall i, j : (f_i = f_j))$.

Many of the traditional approaches to build histograms require a user-defined parameter k, the number of bins. Multiple rules were suggested in the literature to define it based on the number of observations n (e.g. the Sturge's rule: $k = 1 + \log_2(n)$ [Sturges, 1926]). However, it is not suitable for large values of n (i.e.$n > 200$). How can we maintain our histograms representative of the data distribution of a random variable represented by a stream of values? In this Section, we will detail three techniques to maintain the histograms up-to-date on these kind of environments: 1) the K-buckets Histograms [Gibbons *et al.*, 1997], the PiD algorithm [Gama and Pinto, 2006] and 3) the Exponential Histograms [Datar *et al.*, 2002].

3.2.1 K-buckets Histograms

Let $F = \{f_1, ..., f_{k-1}, f_k\}$ be a series of frequency counts of a given event over a random variable X. Let the boundaries be defined by the following set of breaking points $B = \{b_1, ..., b_{k-1}\}$. Gibbons *et al.* [1997] proposed an algorithm to incrementally maintain histograms with a fixed number of bins: k (which is previously defined). Firstly, this method consists into defining two thresholds (one maximum and one minimum) for the frequency in each bin. Secondly, two rules are build to update the buckets (i.e. bins) size: 1) whenever a frequency count f_i on a given bucket $[b_i, b_{i+1}]$, goes greater than the maximum threshold, it is *split* in two by creating a new breaking point between its two previous boundaries b_i, b_{i+1}. The next two buckets are *merged* by removing the breaking point b_{i+2}. 2) In other hand, if such frequency count f_i on a given bucket $[b_i, b_{i+1}]$ goes lower than the minimum threshold, it is *merged* with a neighbor bucket (i.e. the one containing the lowest frequency). The bucket containing the largest frequency is then divided into two.

3.2.2 Partition Incremental Discretization (PiD)

This algorithm was firstly proposed by Gama and Pinto [2006]. It extends the base idea introduced on the K-buckets histograms by removing the constraint of maintaining a static number of bins k. It is reasonable to assume that as more data about a given problem becomes available, it is better to keep more detailed information to describe it. Such level of detail can be easily achieved by *shrinking* the bins width through progressive increases of the number of bins k. Similarly, it may also be useful to reduce this detail on other situations.

To possess such adaptive characteristics, the PiD algorithm maintains two distinct layers: one which runs the K-buckets algorithm using only a maximum frequency threshold (i.e. without merges) and a second one with a dynamic bin width. The first layer contains an user-defined number of bins k_1 which should be considerably lower than any possible number of desired bins k (i.e. $k_1 \ll k$). Then, the second layer is constructed on demand using the parameter k. It basically works by summing up the bins on the layer 1 to achieve an equal-width histogram of k bins. Consequently, $k = c \times k_1 : c \in \mathbb{N}, \forall k, k_1$.

3.2.3 Exponential Histograms

One of the most common data streams consists into timestamped series of 0's and 1's which may refer to a certain event. The idea is to build an histogram capable of counting the number of 1's within a certain sliding window of size N (a user-defined parameter). But what happens if the N is too big for the resources available (i.e. memory space)? The exponential histogram strategy presented by Datar *et al.* [2002] consists of using non-equally sized bins to hold the data. The histogram is composed by a series of buckets and two additional variables: $LAST$ and $TOTAL$. Besides a frequency count, each bucket has a timestamp associated with it. The variable $LAST$ keeps the size of the last bucket while the $TOTAL$ stores the buckets' total size.

When a new 1-type data element arrives, we create a new bucket of size 1 with the current timestamp and we increment the variable $TOTAL$. As long as new time series values are known, two mechanisms are employed to reduce the amount of information kept in memory: 1) to *merge* buckets and 2) to *forget* them. The 1) *merge* operation consists on merging the two oldest buckets of the same size whenever there are $|1/\epsilon|/2 + 2$ or more buckets of the same size (where ϵ is a user-defined parameter which represents the admissible relative error). If the last bucket is merged, we update the $LAST$ variable. The 2) *forget* mechanism depends on the parameter N: all the buckets outside this time window are instantly dropped (and the variables $TOTAL$ and $LAST$ are updated according to this operation.

Using such methodology, we are able to maintain an approximate histograms from streams of data. The estimate of 1's in the sliding window T is given by the following equation

$$T = TOTAL - LAST/2 \tag{3.3}$$

Datar *et al.* [2002] demonstrates that using this methodology for the basic counting problem opens the possibility of adapt many other techniques to work for the sliding window model - not only to maintain approximate histograms but also hash tables or statistics, for instance. In the next Section, we briefly review the techniques commonly applied to time series analysis.

3.3 Time Series Analysis

3.3.1 Definition, Trend and Seasonality

We consider as a time series a sequence of numerical values which represents the evolution of a given random variable over time. Each one of this values has a timestamp associated which represents its order in the sequence. They can be either continuous or discrete.

The majority of the time series patterns can be described in terms of *trend* and *seasonality*. The *trend* represents a general but systematic component (linear or nonlinear) that evolves over time while the *seasonality* represents a periodic repetition of these patterns over time. Fig. 3.1 represents the time series regarding taxi-passenger demand over a month in the city of Porto, Portugal. It clearly exhibits seasonal patterns (i.e. weekly), where the weekends have a lower demand than the work days.

In this Section, we will briefly revise some important methods to deal about the *trends* and *seasonalities* underlying in a time series process. Secondly, we

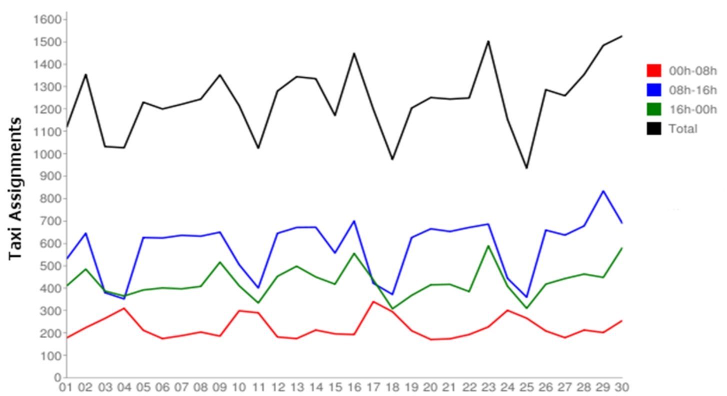


Figure 3.1: Taxi-passenger demand over a month in the city of Porto, Portugal. The x-axis represent month days while the y-axis is the total number of taxi assignments on each one of the considered time spans.

will revise some methods to perform time series prediction and finally we enunciate two State-of-the-Art techniques to measure the similarity between time series.

Trend

A moving average (MA) is commonly used in time series data to smooth out short-term fluctuations and highlight trends or cycles. We can distinguish between two types of MA methods: 1) *averaging methods* where all data points have the same relevance and *weighted averaging methods* where data points are associated with a weight that strengthens their relevance. There are mainly two relevant *averaging methods*:

- **Moving Average**: The mean of the previous n data points:

$$MA_t = MA_{t-1} - \frac{x_{t-n+1}}{n} + \frac{x_{t+1}}{n} \tag{3.4}$$

- **Cumulative moving average**: The average of all of the data up until the current data point:

$$CA_t = CA_{t-1} + \frac{x_t - CA_{t-1}}{t} \tag{3.5}$$

The second group - weighted moving averages - includes:

- **Weighted moving average**: it has multiplying factors to give different weights to different data points. The most recent data points are the most important ones.

$$WMA_t = \frac{nx_t + (n-1)x_{t-1} + ... + 2x_{t-n+2} + x_{t-n+1}}{n + (n-1) + ... + 2 + 1} \tag{3.6}$$

- **Exponential moving average**: like the previous method, it has weights for each data point. However, they are decaying exponentially rather than linearly as they are applied to older data points.

$$EMA_t = \alpha \times x_t + (1-\alpha) \times EMA_{t-1} \tag{3.7}$$

The exponential moving average has the advantage to not require the maintenance of all data points in memory - it fully depends on the α parameter. However, to find an adequate *alpha* is not be as trivial as it seems.

Seasonality

The autocorrelation is one of the most useful statistics to mine the seasonalities within a given time signal. It is the cross-correlation of a time series with itself. It is commonly used to detect not only the existence of periodic signals but also its periodicity. We can define $r(x, l)$ the Autocorrelation as the correlation between x and $x - l$ where l represents the time lag.

$$r(x, l) = \frac{\sum_{i=1}^{n-1}(x_i - \bar{x})(x_{i+l} - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{3.8}$$

Fig. 3.2 plots the autocorrelation of a signal highly similar to the one presented on the Fig. 3.1. It exhibits a clear periodicity of 12 hours.

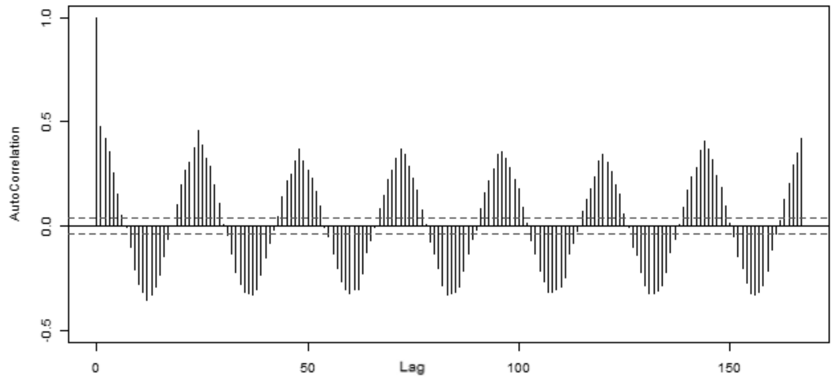


Figure 3.2: Correlogram of the demand on taxi services (13 weeks) obtained from one of the busiest taxi stands in the city (periods of 60-minutes) of Porto, Portugal. The x-axis has the different period lags studied and the y-axis has the correlation within the signal. Note the peaks for each 12h periods.

3.3.2 Time Series Prediction

Typically, a prediction about the next value of a time series is made by mining dependences between time-points. One of the most common ways to define such dependences over time are the *autoregressive* models. The simplest autoregressive model of order 1 is:

$$AR(1) : z_t = \beta_0 + \beta_1 \times z_{t-1} + \epsilon_t \tag{3.9}$$

The simplest method to learn the parameters of AR(1) model is regress Z on lagged Z. If the model is able to mine the dependence structure within the past data points, the residuals are determined without any dependence within. The Autoregressive Integrated Moving Average (ARIMA) is a State-of-the-Art method to perform time series prediction which uses both MA and AR models. It is briefly described in the following Section:

Autoregressive Integrated Moving Average

The AutoRegressive Integrated Moving Average Model (ARIMA) [Box *et al.*, 1976] is a well-known methodology to both model and forecast univariate time series. The ARIMA main advantages when compared to other algorithms are two: 1) it is versatile to represent very different types of time series: the autoregressive (AR) ones, the moving average ones (MA) and a combination of those two (ARMA); 2) on the other hand, it combines the most recent samples from the series to produce a forecast and to update itself to changes in the model. A brief presentation of one of the simplest ARIMA models (for non-seasonal stationary time series) is enunciated below following the existing description in [Zhang, 2003]. For a more detailed discussion, the reader should consult a comprehensive time series forecasting text such as Chapters 4 and 5 in [Cryer and Chan, 2008].

In an autoregressive integrated moving average model, the future value of a variable is assumed to be a linear function of several past observations and random errors. It is possible to formulate the underlying process that generates the time series (taxi service over time for a given stand k) as

$$\begin{aligned} X_t = \kappa_0 + \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} \\ + \varepsilon_t - \kappa_1 \varepsilon_{t-1} - \kappa_2 \varepsilon_{t-2} - ... - \kappa_q \varepsilon_{t-q} \end{aligned} \tag{3.10}$$

where X_t and $\{\varepsilon_t, \varepsilon_{t-1}, \varepsilon_{t-2}, ...\}$ are the actual value at time period t and the *Gaussian white noise'* error terms observed in the past signal, respectively; $\phi_l(l = 1, 2, ..., p)$ and $\kappa_m(m = 0, 1, 2, ..., q)$ are the model parameters/weights while p and q are positive integers often referred to as the order of the model. Both order and weights can be inferred from the historical time series using both the autocorrelation and partial autocorrelation functions as proposed by Box and Jenkins in [Box *et al.*, 1976]. They are useful to detect if the signal is periodic and, most important, which are the frequencies of these periodicities.They are useful to detect if the signal is periodic and, most important, which are the frequencies of these periodicities.

3.3.3 Similarities between Time-Series

Most of time series analysis techniques such as clustering, classification or novelty detection, require to measure the similarity between time series. Let $D(Q, S)$ be defined as a similarity measure between two time series Q, S. A common way to measure it consists of considering some form of *distance* between the two time series. Two of the most commonly used distance metrics on this context are the **Euclidean** distance and **Dynamic Time Warping** (DTW). Such techniques are enunciated and discussed below.

Euclidean Distance

The Euclidean Distance between two time series corresponds to the square-root of the sum of the squared distances from each n^{th} point in the other. Given two time series $Q = q_1, q_2, ..., q_n$ and $S = s_1, s_2, ..., s_n$, the Euclidean distance $D(Q, S)$ can be defined as

$$D(Q, S) = \sqrt{\sum_{i=1}^{n} (q_i - s_i)^2} \tag{3.11}$$

Despite its utility, this metric requires two time series equally sized (i.e. with the same number of elements) to work. It can be quite efficient as a distance but it is not that so as a measure of similarity. For example, if you consider two identical time series, one slightly shifted along the time axis, you will notice that this distance will consider them very different from each other. A distance metric that tries to solve some of this limitations is the DTW [Chu *et al.*, 2002], presented below.

Dynamic Time Warping

Let Z_n and Q_m be two sequences having the lengths n, m, respectively, where n may not be equal to m. If the aim is to align them using DTW, it is necessary to construct an *n-by-m* matrix containing the distances between all points in the two series. Then, a warping path is defined. This warping path is a contiguous set of matrix elements that defines a possible and optimized mapping between Z and Q. The u_{th} element of the warping path is defined as $w_u = (i, j)$, and therefore we have $W = \{w_1, w_2, ...w_k, ..., w_U\}$, which requires the validity of the following in-equation:

$$max(m, n) \leq U \leq m + n - 1 \tag{3.12}$$

This path is subjected to three major constraints:

1. **Boundary conditions**: $w_1 = (1, 1)$ and $w_u = (m, n)$. This requires that the warping starts and ends in the diagonally opposite cells of the matrix.
2. **Continuity**: Let $w_u = (a, b)$. Then $w_{u-1} = (a', b')$, where $a - a' \leq 1$ and $b - b' \leq 1$. This restricts the possible steps in the warping path to adjacent cells (including diagonally adjacent cells).
3. **Monotonicity**: Let $w_u = (a, b)$. Then $w_{u-1} = (a', b')$, where $a - a' \geq 0$ and $b - b' \geq 0$. This forces the points in W to be *monotonically* spaced in time.

In order to build an optimized path satisfying the conditions above, it is necessary to minimize the warping cost:

$$DTW(Z, Q) = min \left\{ \frac{\sqrt{\sigma_u^U W_u}}{U} \right\} \tag{3.13}$$

3.4 Ensembles on Data Streams

The term *ensemble* is used to identify a set of predictor models (for instance, classifiers, regression models or time series analysis ones) for which individual decisions are in some way combined (typically, by voting or by weighting their outputs) to predict/classify novel time/data points [Dietterich, 1997]. The main idea behind any ensemble model is based on the observation that different learning algorithms explore different representation languages, search spaces and evaluation functions of the hypothesis. How can we explore such differences? Specially in the context of dynamic streams, where the target concept may evolve over time?

In this Section, we will discuss two distinct types of ensembles: 1) Sampling techniques to select the best learners on the current data distribution are presented in the Section 3.4.1. 2) in the Section 3.4.2, we will discuss methods which employ the most typical way of ensembling learners from data streams - by a linear combination of their outputs.

3.4.1 Sampling from the Training Set

In this Section, we review techniques to combine different prediction models generated by a single algorithm. Most of these strategies consist into manipulating the training set to generate multiple hypothesis. Typically, the same algorithm is trained with distinct and disjunct distributions of the training data - producing multiple predictive models. Then, these models to classify new examples and their output is somehow combined - typically by some voting technique. This is specially efficient for *unstable* algorithms - their output experience huge changes even with small fluctuations of the training data.

Traditional Bagging

Bagging is one of the most effective techniques for variance reduction. The basic idea firstly proposed by Breiman [1996] consists into producing N replications of the training set by sampling. The following algorithm description is focused on classification problems. Each replication of a training set with m examples is equally sized. The replications may not contain all examples of the original training set but can contain some repeated examples. This technique was originally called *bootstrap aggregation* - and each replicated training set is called a *bootstrap replicate*. The probability p of a given example be selected is given by the following equation.

$$p = 1 - (1 - 1/m)^m, \lim_{m \to \infty} p = 1 - 1/\epsilon \tag{3.14}$$

Each bootstrap contains, on average, $1/\epsilon$ of duplicated examples. All classifiers are then used to classify each example in the test set by using a voting scheme. As described here, this technique requires to perform N random draws from the original training set to produce N *bootstraps replicates*. Therefore it requires a prior knowledge over the entire training set (i.e. a finite one). Such requirement is not compatible with *neverending* data streams, where the data examples are constantly arriving in a unbounded manner.

Online Bagging

Oza [2001] proposed a way to adapt the traditional Bagging (i.e. batch) algorithm to open-ended data streams. A base model is trained with k copies of each one of the m available training examples where the probability mass function of k is given by the following equation

$$Pr(k) = \frac{\epsilon^{-1}}{k!} \tag{3.15}$$

Whenever an example x, y becomes available, each one of the N models is updated using k repeated instances of x, y. k is randomly chose according $k \sim Poisson(1)$. If you notice, the equation 3.15 refers to the probability mass function of a Poisson process where $\lambda = 1$ (i.e. the binomial distribution of k tends to be a $Poisson(1)$ process as the number of available processes tend to ∞). This is a described as a good approximation of the batch learning since the bootstrap training sets generated this way have a similar distribution of the batch ones.

Online Boosting

The boosting algorithm - firstly proposed by Schapire [1990] - proposes to convert a set of weak base learners into a strong one. It maintains a weight for each example in the training set that reflects its importance. Adjusting the weights force the learner to focus on different examples, creating distinct predictive models. In each one of N iterations, the weights are adjusted according to the performance of the N_i model by increasing the weight of the misclassified examples. Like bagging, the final iteration consist into aggregate the learned classifiers - however, the boosting does it so by employing a weighted voting schema (in opposition to the simple voting using on the bagging).

The online adaptation of this algorithm - also proposed by Oza [2001] - has its foundations closely related to the online Bagging one. Let us consider a set of N predictive models $H = h_1, h_2, ..., h_N$ and two sets of parameters $\lambda_c = \lambda_c^1, ..., \lambda_c^N$ and $\lambda_w = \lambda_w^1, ..., \lambda_w^N$ which represents the sum of the weights of the correctly and incorrectly classified samples, respectively, by each individual classifier h_i.

Given a new example $(\vec{x}, y)$. Its initial weight is $\lambda = 1$. Then, the algorithm iterates for each h_i where $i \in 1, ..., N$. For each h_i, it uses k instances of this new example to update itself - where k follows a Poisson process as $Poisson(\lambda)$. Secondly, h_i classifies $(\vec{x}, y)$. Thirdly, the λ_m^c and λ_m^w parameters are updated accordingly with the accuracy of its classification. Finally, the example weight λ is updated using this two parameters values. This process is repeated for each one of the N predictive models. The reader should consult the Section 4.4 in [Oza, 2001] to more details about this algorithm.

3.4.2 Linear Combination of Predictors

One of the first online learning ensemble methods is the **WinNow** algorithm [Littlestone, 1988]. This algorithm combines the predictions of multiple binary classifiers. Initially, each *expert* (i.e. classifier) is assigned with a weight $w_i = 1$. Whenever the weighted vote misclassifies an example, the weight is multiplied by an user-defined constant $\beta \leq 1$. An extension of this algorithm was presented by the same authors as the **Weighted-Majority Algorithm** (WMA) [Littlestone and Warmuth, 1994]. It basically sums all the weights of the algorithms that votes for the same classes. The class with the highest weight sum is our label prediction. Again, the weight attached to wrong predictions is multiplied by β. Such sequential learning forces that the series of values of the weight of any expert in WMA always decreases (i.e. $\beta \leq 1$). This presents a disadvantage in time-changing streams. One of most used strategies to minimize this issue consists of normalizing the weights after each update. However, other algorithms consider update strategies that are more reactive than these ones. One of this algorithms is Weighted Classifier ensemble [Wang *et al.*, 2003]. This model can be adapted to the majority of the prediction problems (classification, regression, time series analysis, etc.). However, the definition below is considered to a time series analysis problem.

Consider $M = \{m_1, m_2, ..., m_z\}$ to be a set of z predictors of interest to model a given time series and $F = \{f_1, f_2, ..., f_z\}$ to be the set of forecasted values to be the next data point on the time series. The ensemble forecast E_t is obtained as weighted average of the outputs F. The weight set $W = w_1, w_2...w_z$ is calculated as $w_i = 1 - e_i$ where $0 \leq e_i \leq 1$ is the error exhibited by the *expert* f_i on the last H data points. H works as an user defined parameter which delimits a sliding window where the predictors are evaluated to perform their output combination for next data point. As long as H decreases, the predictive model becomes more reactive to changes on the models performance.

Nevertheless, evaluate a model performance on a data streams context may be a tricky problem by it own. We briefly revise some of these methods in the next Section.

3.5 On Evaluating Streaming Algorithms

Evaluating Streaming methods is not a trivial task due to three key characteristics: 1) the existence of a continuous flow of data instead of a finite training set; 2) the evolution of decision models over time; 3) data is generated from non-stationary data distributions. Therefore, the approach we must follow to evaluate these kind of predictive models must be based on *sequential analysis*. Sequential analysis refers to the body of statistical theory and methods where the sample size may depend in a random manner on the accumulating data [Ghosh and Sen, 1991].

In this Section, we firstly introduce the two main ways reported in the literature to evaluate methods that learn from data streams. Then, we will briefly introduce some State-of-the-Art metrics based on predefined bounds over loss functions. Thirdly, we will present how to handle the error in time series analysis problems and with non-stationary data distributions.

3.5.1 Basics of Evaluation Metrics

A loss function is a function that maps an event or values of one or more variables onto a real number which represents some "cost" associated with the event. Typically, in our context, the event in question is some function of the difference between estimated and true values for an instance of data. The majority of times, loss functions are employed to evaluate the learners performance. However, the unique characteristics of continuous flows of data require a specific experimental setup to work. There are two main ways to evaluate a learning model on such context:

1. To maintain an independent test set. We can apply a decision model of interest to such test set on a fixed time interval or each p number of examples. The loss accumulated by such tests can be used to monitor the evolution of the model performance.

2. Predictive Sequential: *Prequential* [Gama *et al.*, 2013] where the error of a model is computed from the *sequence* of examples. For each example in the stream, the decision model makes a prediction based only on the example attributes. The prequential error S_i is calculated using the accumulated sum of some loss function of interest L. L uses as input the prediction y and the observed value x as described in the equation below.

$$S_i = \sum_{j=1}^{i} L(x_j, y_j) \tag{3.16}$$

One of the main advantages of this prequential framework is that it does not require to know the true value y in every data point. It can be employed in situations of limited feedback (i.e. by using just the points where y_i is known).

3.5.2 Typical Bounded Evaluation

Considering a prequential framework of evaluation, we can define M_i as the mean loss by the following equation: $M_i = \frac{1}{n} \times S_i$ - independently on the loss function L. Thereby, we can estimate a *confidence interval* for the probability of error, $M_i \pm \varepsilon$, using the Chernoff bound [Chernoff, 1952]:

$$\varepsilon_c = \sqrt{\frac{3 \times \bar{\mu}}{n} ln(2/\delta)} \tag{3.17}$$

where δ is an user-defined confidence level and n are the number of available examples. If we are using a **bounded** loss function - like the 0-1 loss function (see eq. 3.19) - the Hoeffding bound [Hoeffding, 1963] can be used:

$$\varepsilon_h = \sqrt{\frac{R}{2n} ln(\frac{2}{\delta})} \tag{3.18}$$

where R is the range of the random variable. Both bounds are independent of the distribution of the random variable. One of the most typical (and simple) loss functions associated with classification problems is the 0-1 loss function. Let L^i_{0-1} be the 0-1 loss function relative to example i on the data stream. Let x_i and y_i be the predicted and the real label of such example. We can define L^i_{0-1} as follows.

$$L^i_{0-1} = \left\{ \begin{array}{ll} 0 & if x_i \neq y_i \\ 1 & if x_i = y_i \end{array} \right\} \tag{3.19}$$

3.5.3 On Measuring the Error on Continuous Event Time Series

So far, we described some generic evaluation frameworks for stream environments - specially focused on classification problems. However, in many problems the random variable is bounded on a continuous or discrete domain - rather than a nominal one. Whenever we face a regression or a time series analysis task, other type of metrics are employed.

One of the most common is the quadratic loss function $\lambda(x)$ - defined as follows:

$$\lambda(x_i) = C(y_i - x_i)^2 \tag{3.20}$$

where C is an user defined constant. However, other types of errors exist - like an accuracy or an error rate. Three of the most well known methods to do so are the (1) Symmetric Mean Percentage Error$sMAPE$, the (2) Mean Absolute Error MAE and the Root Mean Squared Error $RMSE$. They are usually described as follows[1]:

$$sMAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|x_i - y_i|}{x_i + y_i} \tag{3.21}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i| \tag{3.22}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i|)^2} \tag{3.23}$$

[1] The reader can consult the Section 5.8 in [Witten and Frank, 2005] to know more about such evaluation of numerical predictions.

However, such metrics may not be that adequate when we are facing **non-stationary distributions** of data. Non-stationarity or **concept drift** means that the concept behind the data generation may shift from time to time. Discarding loss of generality, we may evaluate a predictive model using distinct type of metrics such as probability of false alarms or delay detection. In this case, change detection techniques such as Page-Hinkley test may be useful (the reader can consult the Section 5.3.4 in Gama [2010] to know more about the appliance of this algorithm to this specific context).

Part II

Mass Transit Agencies

Chapter 4

Validation of Bus Schedule's Coverage

This Chapter is focused on improving a relevant step of the Operational Planning: the **Schedule Planning**. In Section 2.1.2, we already discussed the lack of relevant works on evaluating the suitability of the number of schedules, as well as their day coverage, to the network' behavior. These stages are crucial because they are highly influential on the further steps of the Schedule Planning process - such as the trips' definition. Moreover, it is well known that a reliable schedule is a key factor to maximize the profitability of the mass transit companies by increasing the passengers' satisfaction [Strathman *et al.*, 1999; Ceder, 2002] and also by reducing the road congestion levels [Schrank *et al.*, 2012].

A Schedule Plan (SP) consists of a set $\mathcal{S} = \{S_1, S_2, ..., S_k\}$ of k schedules which provide detailed information about every trips running on the previously defined routes. Each schedule is associated with a timetable $t_i : i \in \{1, ..., k\}$. Different routes may have different timetables. Nevertheless, they share the number k of schedules and the day **coverage** of each schedule (this should be common to every bus line to help the customers to easily memorize the SP). A definition of the day **coverage** C_i in a given schedule S_i is presented below.

Let $D = \{d_1, d_2, ..., d_s\}$ be a set of s days of interest to include in a schedule plan (typically, $s = 365$ is used - it corresponds to a one year period). The day coverage C_i of a given schedule S_i is represented by the set of days where its corresponding timetable t_i will be followed. It is possible to define it as:

$$C_i = \{d_1, d_2, ..., d_{\theta i}\} : \bigcup_{i=1}^{k} C_i = D \wedge \theta i > 0 \tag{4.1}$$

where θi is the number of days *covered* by the timetable t_i. The set of every schedule day coverages $\mathcal{C} = \{C_1, C_2, ..., C_k\}$ is called *Schedule Coverage*. An illustrative example of that is displayed in Fig. 4.3.

Once established, it is expected that an SP meets the passengers' demand by *following* their mobility needs (namely, their mobility routines). However, today's urban areas are characterized by a constant evolution of road networks, services provided and location (for instance, new commercial and/or leisure areas/facilities). Therefore, it is highly important to automatically assess how the SP suits the needs of an urban area. An efficient evaluation can lead to important changes on a SP. These changes will lead to: (1) a reduction in operational costs (for instance, by reducing the number of daily trips in a given route) and/or (2) a reliability improvement in the entire transportation network, which will increase the quality of the passengers' experience and, therefore, the number of costumers [Yan and Chen, 2002]. Departing from the previous definition of the steps required to build an SP, it is possible to divide the evaluation into two different dimensions: (1) the suitability of the number of schedules k and of the set of their day coverages $\mathcal{C}$ and (2) the reliability of their timetables $\{t_1, ..., t_k\}$ (to test whether the real arrival times of each vehicle at each bus stop are meeting the previously defined timetable). Hereby, we are focused on the first dimension.

To perform such evaluation, two relevant assumptions are stated:

Assumption 4.1. *Days with similar profiles* ***should*** *be* ***covered*** *by the same timetable, which means that they must be included in the same schedule.*

Assumption 4.2. *The number of schedules to use (k) is already* ***known***[1]*;*

Theoretically, all the days *covered* by the same timetable have *exactly* the same daily profile due to the fact that they share the same departing/arrival times. However, the real values of such times (given by the historical AVL data) may *differ* from the original ones. This Chapter describes a framework that explores such differences by *grouping* each one of the days available $d_j, j \in \{1, ..., s\}$ into one of the possible coverage sets, $C_i, i \in \{1, ..., k\}$. This grouping is made according to a distance measured between each pair of days where the criteria rely on their profiles. As output, rules about which days should be covered by the same timetables are provided. Such rules can be used by the operational transportation planners to evaluate whether the current coverage is still meeting the *network behavior* (that is, the real departure and round-trip times). It also provides insights on how can the current coverage be changed in order to achieve that.

The remainder of this Chapter is structured as follows: Section 4.1 describes the data acquisition process and its preparation in detail. Section 4.2 formally describes the approach to this problem and its main contributions to the existing literature. The third Section describes the Experimental Setup used and the results obtained. Such results are discussed in detail along Section 4.4, firstly (1) by highlighting the most relevant patterns and (2) by suggesting a possible Schedule Coverage to meet such constraints. Then, (3) by discussing the possibilities of deploying such methodology on a real world company and by quantifying its impact in our case study. Finally, conclusions from the work hereby described are drawn.

[1] The selection of the number of schedules is not within the scope of this thesis - check Section 2.4

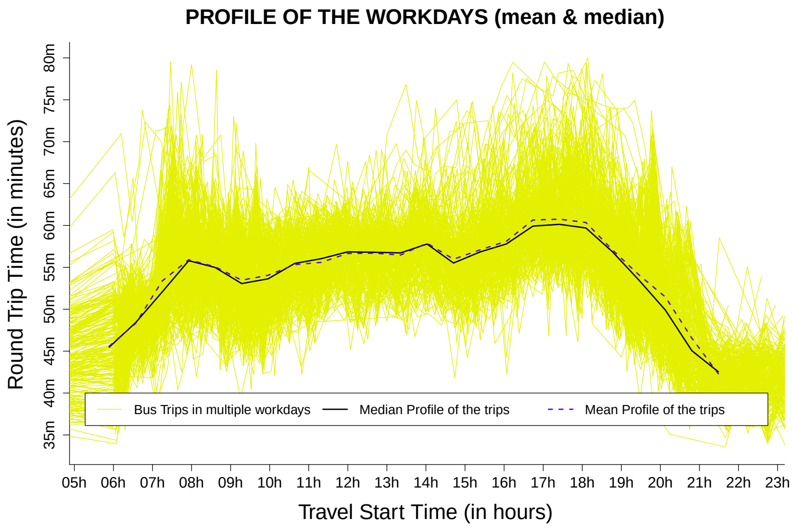


Figure 4.1: Daily Profiles of the behavior of a given route on the working days during a one-year period. The black line represents the median of those profiles while the blue one represent the mean.

4.1 Data Preparation

The case study in this work was the STCP (*Sociedade de Transportes Colectivos do Porto*), the main mass public transportation company in Porto, Portugal. The STCP has a total of 51 lines operating with their own resources. Their AVL system collects information on the location of each vehicle running every 30 seconds. Then the data is sent to the main server.

4.1.1 Data Collection

This study was conducted using a heterogeneous group of four lines - corresponding to six routes - that are representative of the entire network behavior by including all the three possible route types: circular, urban and non-urban routes. The data was collected during a one year-period from January to December 2007 (365 days). The selected bus lines were the 300, 301, 205 and 505. All four lines pass by the *Hospital São João* (HSJ), an important bus/-light train interface in the city. Lines 300 and 301 are arterial urban circular lines, each one corresponding to one route. These lines are quite similar, but with opposite directions and they connect the city center to the HSJ, passing by another important bus/light train/train interface, which is the *São Bento* train station. Lines 205 and 505 both have two routes each: outward and return. Line 205 follows almost the entire peripheral road that marks the city limits, crossing several entrances to the city and several mass transport interfaces, such as Campanhã, which is the main train station. Finally, line 505 serves a suburban area, connecting Porto to a neighboring town, where there is a sea port. The line ends at the HSJ.

An illustration of these routes on the road network in the urban area of Porto is displayed in Fig. 4.2. The orange dots represent the bus stops of each route.

4.1.2 The Schedule Plan in Place

In any SP studied, it is necessary to detail particular seasons that are important to the framework due to their impact on the passengers' behavior. In this case study, these seasons are (1) Easter time (ET), (2) Christmas time (CT) and (3) School Holidays (NSP). The ET represents the period contained in the first eight days of April and the CT corresponds to the last nine days of December. The NSP was set as the period between 15 July and 15 September (including these two boundary days).

The SP at the STCP had a total of four schedules (i.e. $k = 4$) during the year of 2007. Their Schedule Coverage was arranged as follows: *Schedule 1*: Saturdays; *Schedule 2*: Sundays and Holidays; *Schedule 3*: working days during school holidays; *Schedule 4*: working days outside school holidays. Fig. 4.3 illustrates the Schedule Coverage.

4.1.3 Preprocessing

The data was firstly collected for a PhD study and extensively treated and prepared. This is described in detail for a specific route (78-1-1) in sections 2.5.2 and 5.1 of thesis [Mendes-Moreira, 2008]. A similar process was conducted to

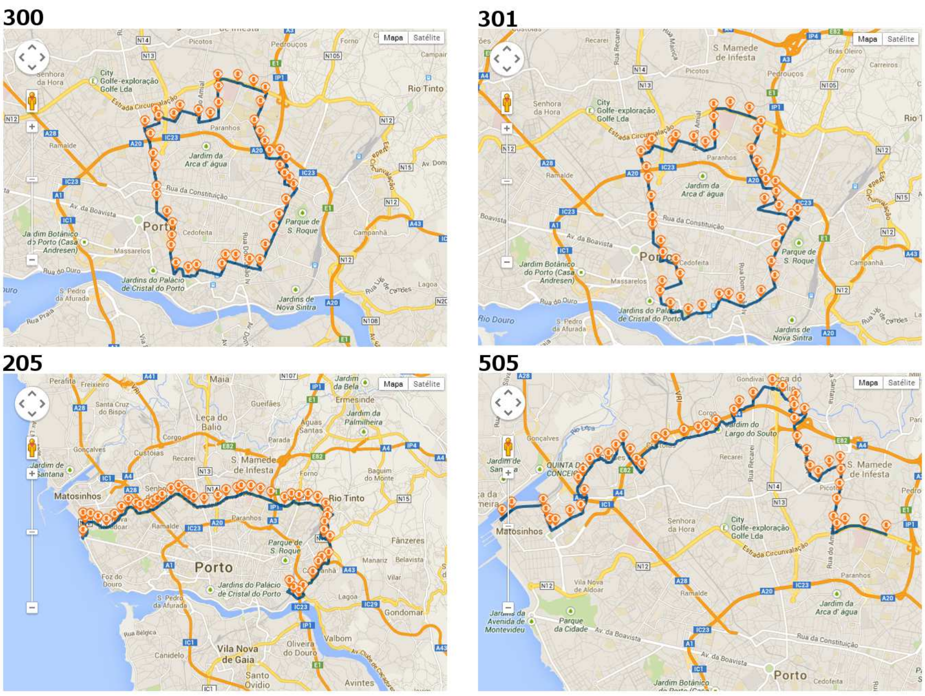


Figure 4.2: Illustration of some routes (one per line) considered over a geographical representation of the road network in Porto, Portugal. Image obtained from [STCP - Sociedade de Transportes Colectivos do Porto, 2013].

obtain the present data and it is briefly described below.

The fleet is equipped with differential GPS devices able to communicate each vehicle's position to the AVL data server. This information is automatically sent to the data server in real-time using General Packet Radio Service (GPRS). The relevant trip data is stored in two different tables from the AVL data server: trip starting time and trip ending time. Obtaining the trip data is not a direct process due to the lack of a primary key identifying each trip individually on the server's database. It is necessary to (1) sort the data and then (2) match pairs of trips starting/ending times, thus making it possible to obtain the round trip times. The data (1) sorted using the timestamps of vehicle's location associated to each trip. The pairs were matched by identifying the records containing each trip's beginning/ending - consequently, it is possible to compute the respective round trip times. Using these times, it is possible to build route datasets. Each dataset has one entry for each trip containing the following information: the starting date of the trip, the departure time, the bus model, the code of the driver, the code of the route service, the day of the year, the type of day (normal day, holiday and floating holidays) and duration of the trip.

As part of the preprocessing task, new datasets were constructed based on the original set. We did so because the original database has some missing values and also an excessive amount of information regarding this specific task. The new dataset contains only the day, the week day, its type and an ordered sequence of round-trip times for the trips completed during the day. The first three variables are used to address the coverage details, while the ordered sequence of round-trip times is used to define groups of days with similar profiles of round-trip times.

Some route values are missing (64 days in 365 $\times$ 6 days possible - see Table 4.1) due to the lack of pair matching and/or other communication failures. To overcome this issue, the *expected* round-trip time profiles were calculated. An

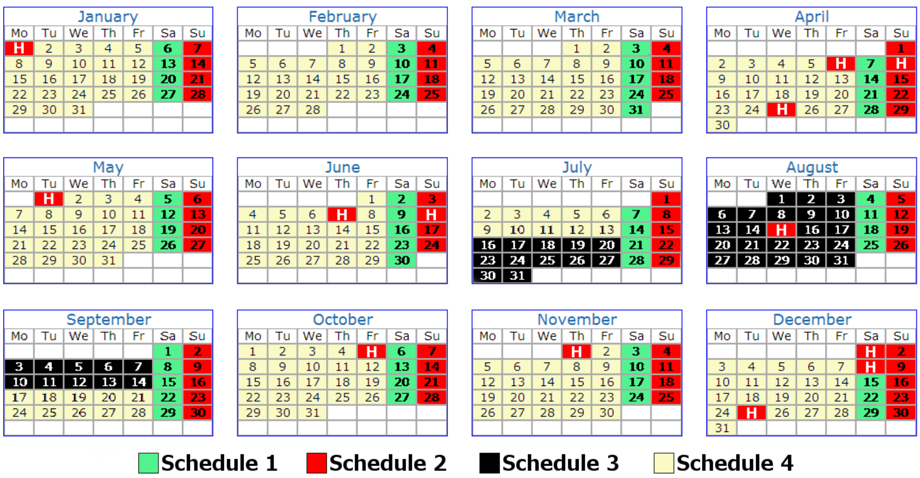


Figure 4.3: Schedule Coverage in place in the case study. The H-symbols represent holidays.

example on these profiles are the light yellow curves in Fig. 4.1. Such curves represent round-trip time profiles of multiple days calculated from a route of interest using the data of the remaining days.

The computation of *expected* round-trip time profiles for the days with missing data consisted of firstly (1) selecting data from the same route but from other days with the same type (for instance, if there were missing data about a certain Tuesday, the information about other Tuesdays would replace it). Then, (2) an *expected* round-trip time profile is built by using both (2a) the number of trips of the most recent similar day (that is the last Tuesday) and (2b) the round-trip times of every past similar days. This preprocessing method forms an *expected* profile for a day with missing data by calculating averages of (2b) these round-trip times into a number of bins equal to (2a) such number of trips. The error introduced by such interpolation method is not significant since the percentage of missing days in every route (2.9% per route on average) is not sufficient to change the output rules that defines the Schedule Coverage in place (which would need, in general, a larger support in the input dataset).

4.1.4 Data Description

Table 4.1 presents a summary of the data used. The columns are the six routes denominated by a XXX_Y mask, where the XXX corresponds to each line and the Y corresponds to the direction considered. The table rows correspond to (1) the total number of trips considered in each route and (2) the number of days with missing data - all in the period considered; 3,4) the maximum/minimum number of daily trips (i.e. DT, in number of trips) in the same period; 5,6) the maximum/minimum travel time (TT - round trip time, in seconds) ever registered for a trip on such period; (7) the median, (8) the mean and, finally, (9) the coefficient of variation of the travel time.

It is possible to observe that line 205 presents a larger number of trips than any other route considered. Lines 300 and 301 present larger round-trip times than the other lines. All the lines present approximate Coefficients of Variation (i.e. the std. dev. of such coefficient from route to route is only $\sigma = 0.0049$). This index can be faced as a *relative* Standard Deviation which exhibits the TT relative variability on each route. These results suggest that such variability is similar from route to route. Yet, it is not possible to infer more than this based only on such coefficients.

4.2 Methodology

The validation framework is divided into three simple steps: firstly, (1) the running times are extracted from the AVL data of just one route and clustered to obtain the *optimal*[2] day coverage for this specific route (each cluster will correspond to a possible schedule). This step is repeated by every route of interest. Secondly, (2) the Schedule Coverage of each route is assembled to

[2] Throughout this Chapter, the individual coverage obtained to each route is referred as *optimal*. In Machine Learning, the concept of optimality often refers to a given fitting process between a supervised learning algorithm and a given output. This is not the case of this particular study, where the optimality concept refers to a schedule coverage which is specifically tuned for a given route.

create a consensual cluster that is common to every route in the network, using consensual clustering techniques. Finally, (3) rules are extracted, obtaining a new SP day coverage (a feasible and readable coverage plan for the entire network). These steps are described below.

Step 1 starts by clustering the day profiles (extracted from a given route) into a predefined number of k schedules/clusters. The days in each group will then indicate the coverages $C_i : i \in \{1, ..., k\}$ for an SP running on a specific route. Since each route data will produce different partitions (for instance, different day coverages), this specific clustering framework is only able to produce an individual analysis to one route at a time, which will correspond to the *optimal* coverage for that single route of interest. Nevertheless, it is not acceptable that each route has its own Schedule Coverage. Consequently, it is needed to find some *consensus* between such route-based partitions in order to increase the applicability of such clustering framework.

In **Step 2**, a consensual day coverage for the schedule network is mined from the partitions extracted from distinct routes. This is done using a well-known consensual clustering technique [Monti *et al.*, 2003]; finally, in **Step 3**, rules are extracted from the consensual clusters obtained and compared to the existing plan. The aim of this step is to turn the resulting clusters into rules which are easily understandable by a larger audience. To do so, a rule induction algorithm was used: the RIPPER [Cohen, 1995].

Naively, the proposed framework can be seen as a hypothesis test having as null hypothesis the fact that the current SP fits the network behavior (however, it is not possible to state which is its significance). The identified changes are the most critical because the network behavior is already conditioned by the previously defined SP. In the same line, important planning variables, such as passenger demand and timetable arrival times, are not directly considered (even when the coverage in place changes are already affecting the round-trip times and, therefore, the profiles obtained). The evaluation of the timetables and of the number of schedules is not addressed in this thesis. In fact, it is necessary to assume a predefined number of schedules to evaluate the Schedule Coverage using this methodology (go to Assumption 4.2 for more details on this matter).

An illustration of this methodology is presented in Fig. 4.4. A formal definition of the methodology presented here is enunciated below.

Table 4.1: Trip Statistics per Route.

	205_1	**205_2**	**505_1**	**505_2**	**300_1**	**301_1**
Number of Trips	21640	20813	9277	5198	13906	14042
Missing Days	16	14	1	3	26	4
Maximum DT	80	78	37	25	58	59
Minimum DT	6	11	7	4	4	6
Maximum TT	4799	4800	4493	4500	5299	5797
Minimum TT	1842	1828	1602	2085	2165	2278
Median TT	3413	3299	3049	3503	4218	4242
Mean TT	3416.04	3313.06	3130.75	3495.10	4203.55	4344.07
Coef. Variation TT	0.1285	0.1349	0.1427	0.1316	0.1279	0.1326

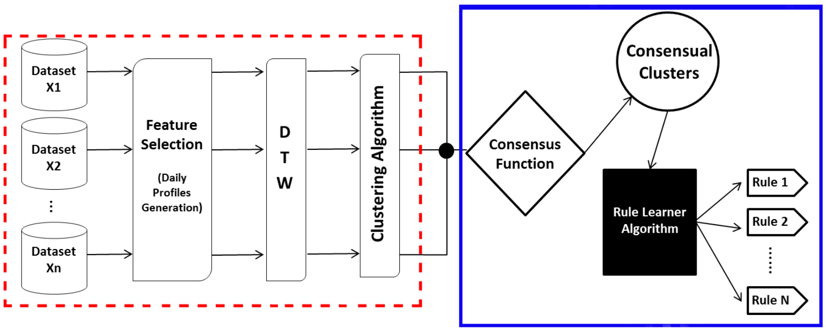


Figure 4.4: Generic representation of the framework. The left red dotted square delimits the step 1 while the right blue square highlights the remaining two steps.

4.2.1 Step 1: How can we find the *optimal* schedule for a single route using AVL data?

Let $X = \{x_1, x_2, ..., x_n\}$ be a set of n datasets (for instance, AVL historical data from n routes) of interest with the same number of samples/trips s. The initial datasets X were firstly turned into new datasets, having each one an entry for each day present in the initial dataset. The information stored per day is a sequence of pairs with the departure time and round trip time. This forms irregularly spaced data sequences (ISDS) of round trip times (i.e. each day has a different number of trips).

Departing from assumption 4.1, we proposed to automatically find **groups** of days which have similar daily profiles based on the AVL data. Such task is known as **data clustering** [Jain, 2010]. A clustering algorithm automatically finds such groups of samples based on a given distance function which estimates the similarity between two different samples. To do it so, a quadratic matrix of distances $s \times s$ is firstly computed. Such matrix maintains the distance between each day based on the ordered series of round trip times (i.e. by the trip departure times). Then, this matrix is the input of a k-dependent clustering algorithm of interest that proposes an ideal SP for that specific route ($P_n = \{C_{1n}, C_{2n}, ..., C_{kn}\}$). However, common distance measures - such as the Euclidean - are very sensitive to variations in both depth and in granularity of the time axis, such as the ISDS used here. To overcome this problem, the use of the *Dynamic Time Warping (DTW)* distance algorithm is proposed. This was firstly proposed by Chu *et al.* [2002] and it was already described in the Section 3.3.3.

4.2.2 Steps 2,3: Finding Consensual Rules to build a Schedule Plan

By partitioning each one of the datasets into k clusters, it is possible to define the resulting non-overlapping subsets of X, denominated P, according to the following definition:

$$P = \{P_{11}, P_{12}, ..., P_{1k}, ..., P_{n1}, P_{n2}, ..., P_{nk}\}, k \geq 2 \wedge k \in \mathbb{N} \tag{4.2}$$

$$\bigcup_{m=1}^{k} P_{im} = x_i, P_{ij} \cap P_{il} = \emptyset, \forall_{i,j,l,k} : j \neq l \tag{4.3}$$

where $\{P_{i1}, P_{i2}, ..., P_{ik}\}$ represents the *optimal* day coverage for a given route i (i.e. the schedule day coverage set $\{C_{i1}, C_{i2}, ..., C_{ik}\}$). By defining P as the k partitions formed from the n input datasets, it is necessary to establish a **new distance measure** between each possible pair of days d_i based on the agreement between the partitions (i.e. a consensual clustering of the data provided by every input routes).

Let $M_i(s \times s)$ (for instance, a quadratic matrix with the number of days considered for each route where each position is set as 1 if the days are in the same schedule and 0 if they are not) be the co-association matrix (or connectivity matrix) representing the clustering membership for the samples in the X_i data set and a given number of partitions k. It can be obtained as follows:

$$M_i(r,j) = \begin{cases} 1 & \text{if } r \in P_{il} \wedge j \in P_{im}, l = m \\ 0 & \text{if } r \in P_{il} \wedge j \in P_{im}, l \neq m \end{cases}, l, m \in \{1, ..., k\} \wedge l, m \in \mathbb{N} \tag{4.4}$$

Then, it is possible to calculate the agreement matrix $\mathcal{M}$ (the consensus between every SP found) and the distance consensus matrix $\mathcal{D}$ using the following equation:

$$\mathcal{M} = \Sigma_{m=1}^{n} \frac{M_m}{n}, \mathcal{D} = 1 - \mathcal{M} \tag{4.5}$$

The resulting matrix $\mathcal{D}$ is a quadratic $s \times s$ distance matrix related to all samples (the distances between all days considered). By applying a k-dependent clustering algorithm of interest to $\mathcal{D}$, it is possible to obtain the dataset $\mathcal{P}$ of k consensual partitions from the datasets in X:

$$\mathcal{P} \equiv clusteringAlgorithm(P, k) \equiv \{\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_k\} \tag{4.6}$$

where each $\mathcal{P}_i : i \in \{1, ..., n\}$ will contain a set of days $\{d_1, ..., d_z\} : z > 0$. Using the consensus function definition described in equations 4.4 and 4.5, it is possible to obtain the consensus clustering for the input datasets. Using these new partitions, logical rules can be extracted using a rule induction algorithm such as the RIPPER [Cohen, 1995]. The base idea is to train a rule-based classifier based on the entire dataset by using each sample's cluster as its own label.

4.3 Experimental Results

This Section starts by describing the experimental setup used in the experiments. Then, the results obtained with the setup are presented.

4.3.1 Experimental Setup

Firstly, the k-Means [MacQueen, 1967] was chosen as a clustering algorithm due to its simplicity, efficiency and efficacy [Lloyd, 1982; Ball and Hall, 1965; Jain, 2010]. To reduce the k-Means random start effects, a deterministic divisive hierarchical clustering was employed, as proposed in [Su and Dy, 2004].

Secondly, both the individual and the consensual clustering experiments were carried out using the R language [R Core Team, 2012]. The k parameter values varied from 2 to 7. This was done because it is not acceptable and/or common to have a number of schedules outside this range [3].

Finally, the J-RIP algorithm - the JAVA implementation of the RIPPER algorithm - was applied to the consensual partitions using the WEKA software [Hall *et al.*, 2009]. A set of seven intuitive decision variables (i.e. features) was used to characterize each day: (1) WEEKDAY: the day of the week {*Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday*}; (2) DAYTYPE: the type of the day ({*holiday, normal, non_working_day, weekend_holiday*} where a non working day represents a working day where the public sector does not work - even if it is not an official holiday); (3) MONTH: $\{1, ..., 12\}$; (4) EASTERTIME: boolean, (5) CHRISTMASTIME: boolean and (6) NONSCHOOLPERIOD: boolean; (7) SCHEDULE: the schedule proposed for each day $\{1, ..., k\}$. Such variables are then used by the RIPPER to output rules that can meet as much as possible the coverage proposed for each schedule using the SCHEDULE variable as target.

The RIPPER outputs a set of rules in a hierarchically divisive form (e.g. like a decision tree based on rules). An accuracy evaluation metric was defined as

$$Accuracy = \frac{\textit{Number of Days Classified Correctly}}{\textit{Total of Days}} \tag{4.7}$$

Typically, an *accuracy* metric is employed in classification problems - which is not our case. Nevertheless, it is employed here to evaluate *how* representative the rule set is of the coverage proposed by the consensual clustering process. This was done by measuring a possible accuracy as *if* the obtained rule set was considered as a classifier (which is the core of the J-RIP algorithm). Comparative tests using the same partitions (i.e. training sets) as test sets were then performed (i.e. each Schedule is considered a possible class (SCHEDULE) and each day is seen as a sample defined by the values of the remaining six features).

The J-RIP algorithm takes four parameters: (1) FOLDS: it determines the amount of data used for the pruning stage[4]; (2) WEIGHT: the minimum total weight of instances in a rule (i.e. it works like a minimum support threshold to consider a rule as meaningful); (3) OPTIMIZATIONS: the number of runs in the optimization process and (4) SEED: a numerical seed used to randomize the data. The following default values were used to this parameter set: 3, 2, 2 and 1, respectively. The purpose on employing RIPPER is to demonstrate that it is able to extract rules (which highlight the patterns underlying on our data) for those who are not familiar with Machine Learning techniques. Consequently, no sensitivity analysis was carried out on such parameter value combination and this value set was used in every experiment conducted.

It is relevant to highlight that this methodology is not an automatic classifier to assign a Schedule to each day. The primary goal of the SP is to meet a certain expected demand minimizing the quantity of resources employed [Ceder, 2002]. However, changes on the Schedule Coverage (e.g. to force the Saturdays to have the same timetable as the Sundays and Holidays) may not be possible due to these previous definitions (e.g. number of drivers and/or vehicles available on Saturdays). This framework should be seen as a decision support tool that

[3] It is desirable to have a number of schedules as low as possible to make it easier for the passengers to memorize the SP [Furth *et al.*, 2003].

[4] The data is divided into multiple folds; typically one of the folds is used on pruning while the others are used to grow the rules.

Table 4.2: Daytype distribution for the consensual clusters.

k=2	**MON**	**TUE**	**WED**	**THU**	**FRI**	**SAT**	**SUN**	**TOT**	**HOL**	**SHO**	**NSW**	**NSH**	**CHR**	**EAS**
1	48	46	47	47	47	1	0	**236**	0	3	44	0	4	4
2	1	3	2	2	2	**48**	**48**	**106**	**12**	2	1	17	6	4
k=3	**MON**	**TUE**	**WED**	**THU**	**FRI**	**SAT**	**SUN**	**TOT**	**HOL**	**SHO**	**NSW**	**NSH**	**CHR**	**EAS**
1	1	3	2	2	2	**48**	**48**	**106**	**12**	2	1	17	6	4
2	19	19	21	22	18	1	0	**100**	0	1	11	0	3	0
3	29	27	26	25	29	0	0	**136**	0	2	33	0	1	4
k=4	**MON**	**TUE**	**WED**	**THU**	**FRI**	**SAT**	**SUN**	**TOT**	**HOL**	**SHO**	**NSW**	**NSH**	**CHR**	**EAS**
1	25	24	21	22	25	0	0	**117**	0	1	13	0	2	4
2	1	3	3	1	2	**49**	**48**	**107**	**11**	2	1	17	6	4
3	12	15	13	15	13	0	0	68	1	1	3	0	2	0
4	11	7	12	11	9	0	0	50	0	1	**28**	0	0	0
k=5	**MON**	**TUE**	**WED**	**THU**	**FRI**	**SAT**	**SUN**	**TOT**	**HOL**	**SHO**	**NSW**	**NSH**	**CHR**	**EAS**
1	14	18	11	11	11	0	0	**65**	0	1	7	0	0	3
2	15	7	14	13	17	0	0	**66**	0	1	9	0	4	1
3	1	2	1	1	2	9	47	**63**	8	1	1	8	4	2
4	18	21	21	23	19	0	0	**102**	0	1	**28**	0	0	0
5	1	1	2	1	0	**40**	1	**46**	4	1	0	9	2	2
k=6	**MON**	**TUE**	**WED**	**THU**	**FRI**	**SAT**	**SUN**	**TOT**	**HOL**	**SHO**	**NSW**	**NSH**	**CHR**	**EAS**
1	13	14	14	12	10	0	0	**63**	0	1	2	0	0	3
2	12	11	10	11	13	0	0	**57**	0	0	11	0	3	1
3	4	0	1	2	2	**47**	**46**	**57**	3	0	3	9	3	2
4	13	10	11	13	17	0	0	**64**	0	2	**24**	0	0	0
5	6	11	11	10	6	0	0	**44**	0	0	4	0	0	0
6	1	3	2	1	1	2	2	**57**	**9**	2	1	8	4	2
k=7	**MON**	**TUE**	**WED**	**THU**	**FRI**	**SAT**	**SUN**	**TOT**	**HOL**	**SHO**	**NSW**	**NSH**	**CHR**	**EAS**
1	6	3	7	9	9	0	0	**34**	0	0	**26**	0	0	0
2	8	7	6	7	10	0	0	**38**	0	0	5	0	1	1
3	2	1	1	0	1	**26**	**19**	**50**	**4**	0	1	7	1	4
4	2	2	1	2	1	**23**	**29**	**60**	**8**	2	0	10	6	0
5	7	6	9	5	5	0	0	**32**	0	1	7	0	0	0
6	11	13	12	11	8	0	0	**55**	0	1	0	0	0	3
7	13	17	13	15	15	0	0	**73**	0	1	6	0	2	0

should be used together with other information, namely, the resources available in each scenario.

4.3.2 Results

The results are displayed in three distinct dimensions: (1) the resulting distribution of days along the clusters is presented in Table 4.2; (2) an illustration of the distribution of days among the $k = 4$ clusters (i.e. the Schedule Coverage) is shown in Fig. 4.5; (3) Fig. 4.6 presents a decision tree exhibiting the rules learned from the consensus clustering using 2 to 4 schedules.

The acronyms used in Table 4.2 can be defined as follows: TOT is the total number of days within the cluster; MON (Monday) to SUN (Sunday) corresponds to the number of days in the cluster by weekdays; the HOL column represents the holidays (including the ones during the weekend), the SHO is the sum of the floating holidays and non-working days; the NSW and the NSH are, respectively, the working days and the weekends during school holidays; the CHR represents the days in Christmas Time and the EAS is the Easter period.

Fig. 4.5 displays the Schedule Coverage provided by the framework presented for a scenario with four Schedules. The x-axis represents the days of the year where the first day of each month is highlighted with an axis caption. The colored points correspond to the days and the colors represent different months of the year. The y-axis are the possible schedules where the days can be grouped. This figure shows seasonalities (i.e. a day of the same type that is grouped in different schedules depending on the month) that are not observable in Table 4.2.

In Fig. 4.6, the circles represent the schedule found in each tree leaf, while the rectangles contain the conditions in each tree node. The left branches should be followed when the condition is satisfied. The accuracy achieved by each set of rules in the three Schedule Coverages considered $k = \{2, 3, 4\}$ were 0.97, 0.78 and 0.77, respectively.

4.4 Discussion

This methodology does not depend on the number of k schedules previously defined, as can easily be observed by the variation of this parameter in Table 4.2. It can be applied to any public transportation network, even if the experiments presented here is considering a case study where only one company is running. For that, it is only necessary to deploy a bus dispatch system whose fleet is equipped with a communication system capable of automatically transmitting (with a certain but short periodicity) the vehicle's position (in GPS coordinates) associated with a timestamp (also known as AVL system).

The amount of data used to conduct these experiments - for one year - may not appear to be sufficient to consider all extracted patterns *meaningful*. However, by observing Table 4.2, it is possible to state that the results are sound because they suggest some relevant differences from the Schedule Coverage in place (please see Fig. 4.3). Its ability to illustrate the similarities between the daily profiles in different routes (even where each route provides heterogeneous insights) is key, especially if we consider that such results already depend highly on the Schedule Plan (number, coverage and timetables) already in place.

There is a main pattern common to almost every number of schedule k considered, which is depicted in Table 4.2: Saturdays and Sundays should use the same timetable. Such conclusion may reduce the number of necessary resources since the Saturdays used their own timetable in our case study (see Fig. 4.3)

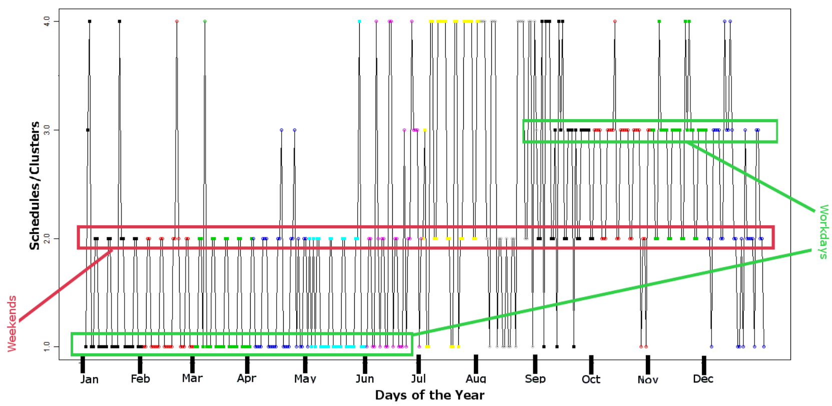


Figure 4.5: Consensual Schedule Coverage proposed for $k = 4$ along the months of the year. The point colors represent their months.

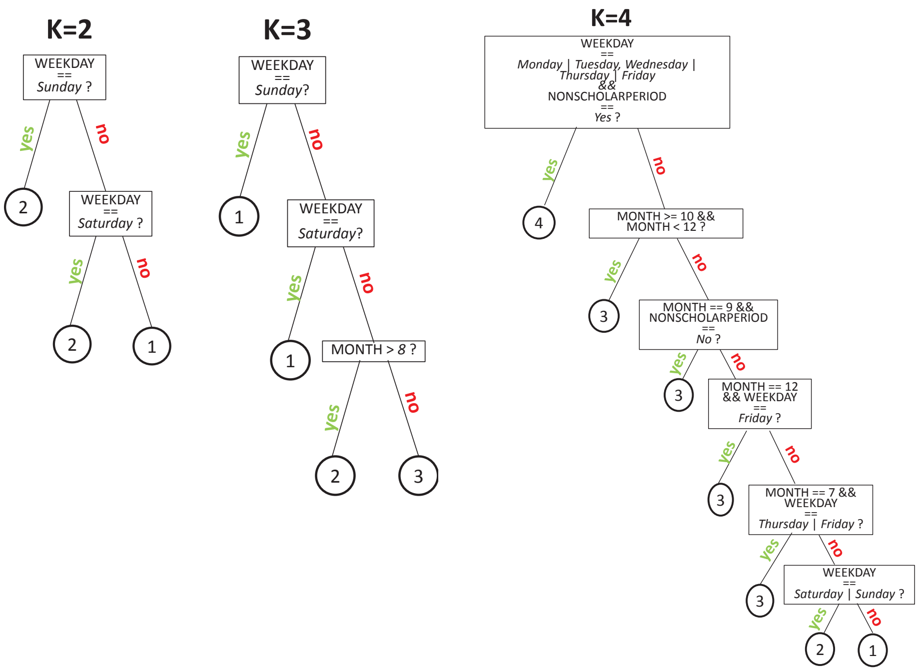


Figure 4.6: Illustration of the rule sets obtained from three consensual Schedule Coverages $k = \{2, 3, 4\}$ as decision trees.

- namely the number of driver shifts, driving hours and/or the necessary vehicles. This happens even for large values of k - see cluster number two in Fig. 4.5. Another pattern confirms that the Non School Period working days should remain with its own individual schedule, while the weekends may be grouped with the remaining Saturdays and Sundays - as already proposed by the SP in place.

A relevant but distinct pattern is observable in Fig. 4.6: the working days in the School Period during the months of September, October, November and December must be put on an individual schedule. This difference is even more visible when we consider the same $k = 4$ schedules that are currently in place in Porto: it is possible to observe a clear difference between clusters one and three in Fig. 4.5. Such difference occurs due to a change in the coefficient of variation of the round-trip times. In fact, they are completely distinct from the remaining working days (i.e. distinct daily profiles). Such differences correspond to round-trip times larger than the usual in some periods of the day. This phenomenon may be explained by the weather conditions in the city of Porto during this period, where storms are frequent, or by some unexpected event, such as long term work on an important city road. However, it is not possible to determine that for sure and the reasons behind this difference are not addressed in this work.

The rules learned can cover the majority of the days considered ($\geq 77\%$), thus demonstrating its capacity to turn the Schedule Coverage obtained into easy-to-read information that cover almost the entire partitioning found by the consensual clustering previously applied.

Therefore, it is possible to confirm the importance of this tool as it provides useful insights on the coverage of the bus schedule. In fact, this framework can find rules that cannot be discovered using the evaluation methods already described in the literature. Such insights can be used to produce a new Schedule Coverage capable of reducing the variability observed between the real and the scheduled round-trip times. A possible proposal to do that in this specific case study is described below.

4.4.1 A Schedule Coverage Proposal

From the analysis of the consensus clusters, five novel constraints to our Schedule Coverage can be drawn:

1. The working days should be in a schedule separated from the remaining days (as suggested by Table 4.2 where this type of days is commonly grouped in schedules that are different from the weekends and/or the holidays one);
2. The working days in a school holiday period should be in an individual schedule (check the values in bold in the NSW column in Table 4.2 for $k \geq 3$ to see some examples of this pattern);
3. The weekends and the holidays should be in an individual schedule (a good illustration of these is made by cluster 2 in Fig. 4.5 or in Table 4.2 - especially for $k < 5$);
4. The CT could be in the same schedule as the weekends and the holidays (typically Christmas Time days, represented by the CHR column in Table 4.2 are grouped with the weekend days or holidays);
5. There is a clear difference between the working days in the last four months and those in the remaining months (visible in clusters 1 and 3 of Fig. 4.5).

Following these constraints, several hypotheses can be made to re-arrange the Schedule Coverage on this case study. However, they must meet other operational planning constraints such as the number of drivers/vehicles available and their shifts [Ceder, 2002]. For more information on this topic, the reader can consult the following survey on urban planning for public transportation companies [Vuchic, 2005].

A possible new Schedule Coverage - according to the current number of schedules - could be the following:

Schedule 1 working days from January 1st to July 15^{th} (beginning of the school holiday period);

Schedule 2 working days from July 15^{th} to September 15^{th} (school holiday period);

Schedule 3 working days from September 15^{th} to December 31^{st};

Schedule 4 all non-working days including all holidays and weekends.

4.4.2 Potential Infusion and Impact

The use of the proposed framework depends on the perception that transportation planners have about its usefulness. In this Section, the main issues on evaluating the changes to the existing SP coverage are described. Then, an approach to measure the usefulness of such framework in a way that could be understood by the planners is presented.

The main obstacle to perform such evaluation is the inexistence of data obtained with both the current and the new schedule plans. In our case study, evaluating a new SP coverage is hardly done before deployment (see Section 2.3.3 to know more about such issue). The main reason for that is that by using a different coverage, various schedules should be used in order to better adjust the schedules to traffic in different days. Despite this difficulty, the proposed approach must be evaluated **prior** to deployment.

Reducing the variance of round-trip times originated by the same scheduled trip has a potential impact on three different components of revenue and costs for a bus company: (1) the revenue can be increased, (2) and the budgeted costs and (3) non-budgeted costs can be reduced.

The first component can happen when there is an increase in client satisfaction as a consequence of the perceived increase in service quality. Measuring such impact is very difficult without generating data based on the new SP. However, it is easier to define a scheduled round-trip time (TT) that is more adjusted to the actual TT when the variance is lower. The method used reduces the variance inside the groups. For this reason, it is expected that the new schedules will improve the passengers' perception of service quality.

The second component is probably the easiest to estimate. In fact, when the variance of TT inside the groups reduces, it is possible to reduce slack times, which have an impact on the definition of crew services, increasing the percentage of driving time in these services. The average cost of the drivers per minute is an important key performance indicator for a public transport company and can be used to estimate the reduction of budgeted costs caused by reducing time $\times$ drivers. This cost is the most important of the budgeted operational costs. It should be emphasized that a small reduction in slack times can cause an important decrease in operational costs due to the increase of the averaged travel time per driver duty.

The third component occurs when it is necessary to adopt extra measures. This happens when there are disruptions between the actual and the scheduled service. The operational planners can create a tighter or wider schedule. In the first case, slack times will be shorter but the probability of disruption increases, thus increasing the non-budgeted operational costs. In the second case, the slack time will be larger, reducing the probability of disruption and, consequently, reducing the non-budgeted operational costs, and yet increasing the budgeted component of the cost. By reducing the variance inside the groups, and maintaining the same probability of disruption, budgeted costs must necessarily fall.

In this case, the method proposed was evaluated by estimating the variance of the trips. This was performed by grouping the trips by route, schedule and scheduled trip start time, and by calculating the sample variance for each group. Then, the global variation was calculated for both the current coverage and the coverage proposed here using the weighted average of the variances in all groups previously described. This weighted average reduces the degrees of freedom in each variance used, as explained in any introductory book on statistics. Comparing the sample standard deviations, the results are inconclusive (present coverage: 594.3 seconds; proposed coverage: 607.8 seconds). It is important to emphasize that these results are necessarily biased by the use of trips generated using the current coverage. Indeed, in such conditions, it is natural that the generated trips are more adapted to the current schedule. This is particularly true for circular routes, as it is the case of lines 300 and 301. However, this result does not invalidate the reasonability of such approach. Its evaluation after deployment, even if in a controlled way (for instance using only a small number of routes) would be particularly important.

This work is now ready to be used. However, its usefulness for a bus company depends on its ability to cover, at least, all functionalities when creating and maintaining timetables. Moreover, since the definition of timetables has an impact on the remaining steps of operational planning (as described in Sect. 2.1.2), this kind of software will be especially interesting when included in Decision Support Systems that cover all steps of operational planning.

4.5 Final Remarks

Most classical approaches to schedule evaluation rely only on how to change the defined timetables and driver shifts. However, these definitions are based on previous definitions which could not be evaluated using an automatic algorithm. Additionally, such changes usually represent an increase in operational costs, for instance, due to increases in the number of running vehicles, slack times and/or driver shifts.

To our best knowledge, this is the very first framework capable of evaluating whether the current coverage fits the network needs. The insights hereby discovered will enhance the operational planning tasks by providing novel decision variables to the planners. These variables carry information that can cause an impact on planning: by optimizing the Schedule Coverage, the planners will be able to take full advantage of the existing resources, or even reduce related costs, while improving the passengers' perception of service reliability and providing SP day coverage according to their mobility needs.

This problem was addressed using a reasonably complex Machine Learning system. The steps used were: (1) k-means with the DTW distance per route to find an optimal schedule coverage for each route based on its trip daily profiles, (2) a Consensual Clustering to find a consensual day partition between all the considered routes, and (3) rule induction using the RIPPER algorithm to extract understandable rules. The use of consensual clustering is emphasized to address an important real problem in the transportation area. The employment of the rule induction system broadens the target audience for this methodology by removing the need for a solid background on Machine Learning techniques.

The experiments were conducted in a specific case study, a public transportation operator in Porto, Portugal, which highlighted the usefulness of this framework: it is capable of extracting important information regarding the Schedule Coverage from a vast amount of data. It is independent of the number of schedules k and, more importantly, of the company where the framework will be deployed (only an AVL communication system is required). We believe that the work presented along this Chapter is unique due to the type of patterns it can reveal about the Schedule Coverage. Moreover, it opens new research lines for evaluating Schedule Plans by broadening its scope to the coverage dimension.

Chapter 5

Online Bus Bunching Mitigation

PT reliability could be defined either in terms of punctuality - the extent to which operations adhere to the planned schedule - or in terms of regularity - the extent to which vehicles are evenly spaced, implying even headways, the time interval between successive vehicles running the same route [TCRP, 2003]. In the case of high-frequency routes (headways of 10 minutes or shorter), regularity is the main indicator of service reliability since it is the main determinant of passengers' waiting time [Cats, 2014]. Headways are inherently instable due to a positive feedback loop between the headway, the number of passengers waiting at the stop, dwell times and successive headways [Daganzo, 2009]. For example, a small bus delay provokes an increase in the number of passengers in the next stop. This number leads to an increase in the dwell time (bus service time at a stop) and consequently, it further increases the bus delay. On the other hand, the next bus will have fewer passengers, shorter dwell times and will gradually catch up the preceding bus. This snowball effect will result with the pairs of buses forming a platoon as illustrated in Fig. 5.1. This phenomenon is denominated as **Bus Bunching(BB)** [Daganzo, 2009; Moreira-Matias *et al.*, 2012b, 2014a].

The prevalence of BB is one of the most visible characteristics of an unreliable service. Two (or more) buses running together on the same route is an undeniable sign that something is going terribly wrong with the company's service. Operational Control can potentially address BB in real-time. This Chapter describes a methodology focused on exploring both historical and real-time AVL data to build automatic control strategies, which can mitigate BB from occurring while reducing the human workload required to make these decisions. It provides a **complete bottom-up methodology**, from fundamental theoretic aspects of capturing the BB process stochasticity to practical issues involved with actions deployment and on the evaluation of their impacts.

The symbols and notations used throughout this Chapter are provided in Tables 5.1 and 5.2. The remainder of it is structured as follows: The Data Collection employed on this study's experiments is described in Section 5.1, along with some details about its preprocessing and our real world Case Study.

Section 5.2 introduces some related work on the BB topic, grounding the contributions of this framework to such State-of-the-Art. The third Section details the proposed stepwise methodology and highlights its main contributions to the State-of-the-Art on this topic. Section 5.4 presents the experimental setup, by introducing some evaluation metrics, a tuning framework to adjust the values of the methodology's parameters, the artificial demand model employed to produce synthetic data about the passenger demand and the experimental results. Section 5.5 presents some discussion about these results and the potential impact of this framework on a real world Control center. Finally, some final remarks are presented along with future research directions on this topic.

5.1 Data Preparation

The real-time framework for detecting and preventing bus bunching is applied to the same case study of the previous Chapter (i.e. STCP). Conversely to the task described on that Chapter, this data was masked due to privacy issues. This BB study was conducted using a heterogeneous group of **nine bus lines** (A-I) - that include both urban and non-urban routes covering different parts of great Porto area. The data were collected during a one year-period from January to December 2010 (365 days). Each line has two route-directions A1, A2, B1, ..., I2.

Line A is a commuter line between downtown and *Vila D'este*, a large poor neighborhood located in the southern edge of the Douro river which trespasses many rural areas. Line B is a major urban line that connects the major city street market to luxurious neighborhood on the city seaside (*Castelo do Queijo*). Line C is also a major urban line between *Viso* (an important neighborhood in Porto) and *Sá da Bandeira*, a downtown bus hub. Line D connects downtown to *Hospital São João* (HSJ), an important bus/light train terminal in the northern part of the city. Line E connects downtown to an highly populated neighborhood in the east (*S.Roque*). Line F is an arterial urban line. It traverses the main interest points in the city by connecting two important street

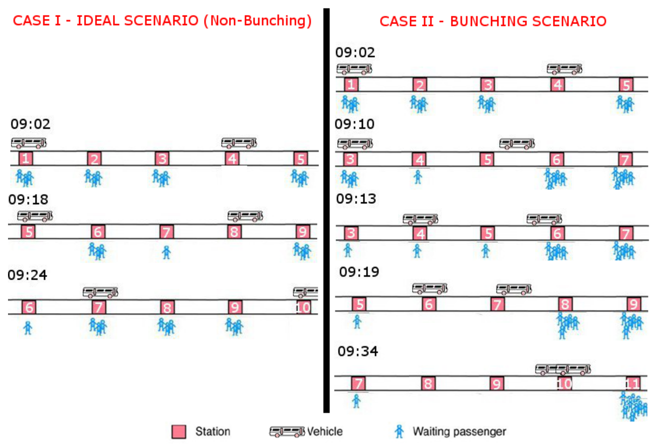


Figure 5.1: Bus Bunching illustration.

markets: *Bolhão* - located in downtown - and *Mercado da Foz*, located on the most luxurious neighborhood in the city. Line G connects the city downtown to the farthest large-scale neighborhood in the region (*Maia*). Line H departs from an important terminal located on the city outskirts (*Marquês*) to a highly dense residential area in the east (*Rio Tinto*). Finally, Line I connects the two major transport hubs in the city: *Trindade*, that joins all the five light tram lines in the city and *Sto. Ovídeo*, which is the southernmost light tram station that provides bus connections to most of the neighborhoods located in this region.

Table 5.1: Notation and symbols about the BB Control Framework employed along this Chapter.

n	total number of trips on the dataset of a given route
$f_{i,j}$	Planned Headway established for a given pair of trips, (i,j)
$H^b_{i,j}$	Observed Headway on a given pair of trips (i,j) at a bus stop b
b_i	i_{th} bus stop of a given route
T_i^j	arrival time of the bus running the trip i to the bus stop j of a given route
$TT_{(i,j)}$	travel time between two bus stops of interest $b_i, b_j : j > i$
$RT_i^{(l,l+1)}$	non-stop run time of the trip i in the road segment between two consecutive stops b_l, b_{l+1}
dwT_i^l	dwell time of a given vehicle/trip i on the bus stop b_l
s	total number of stops of a given route
η	headway-based minimum threshold to consider a BB event between two trips
$LTT_i^{(l,l+1)}$	Link Travel Time between two consecutive stops b_l, b_{l+1}
θ	number of days employed to build the training set to predict the LTTs on a daily basis
Δ_{y_i}	online update made to the previous LTT prediction y_i in place
r_y	residuals of the predictions made to y
α	constant user-defined learning rate of Δ_{y_i}
$\alpha(r_y)$	dynamic residual-based learning rate of Δ_{y_i}
κ^2	constant user-defined learning rate of $\alpha(r_y)$
e	the index of the most recently completed trip
P_e	set of LTT predictions made for the trip e
μ_e	average prediction residual for the completed trip e
ϕ	user-defined maximum threshold for the amount of trip-based residual μ_e
β^2	user-defined residual-based learning rate of $\alpha(r)$ to apply the trip-based update rule
E_c	offline prediction for the headways of the current trip c
$\mathbb{E}_c$	online prediction for the headways of the current trip c
HR_c	residuals of the headways' offline prediction for the current trip c
$H'R_c$	residuals of the headways' online prediction for the current trip c
$\gamma(a,z)^i$	dynamic residual-based learning rate for the stop b_i given the headway's residuals a, z
$[\gamma_{min}, \gamma_{max}]$	user-defined parameters to bound the domain for the learning rate $\gamma(HR_c, HR'_c)$
D^{b_i}	Gaussian p.d.f. of the Headway between two consecutive trips on a given bus stop i
μ_{b_i}	mean value for defining the Gaussian distribution D^{b_i}
σ_{b_i}	Standard deviation for defining the Gaussian distribution D^{b_i}
τ	user-defined sliding window size to compute the recent Variance of $H^i_{k,k+1}$
$p(BB^i_{k,k+1})$	BB likelihood for the pair of trips $\{k, k+1\}$ on the bus stop b_i
$\mathbb{D}^{b_j}$	descendant ordered vector of BB likelihoods for the downstream stops of b_j
BS^{b_j}	BB score to quantify the likelihood of occurring a BB event on the downstream stops of b_j
n^j	number of *agreements* (i.e. positive likelihoods) needed to compute BS^{b_j}
ψ	frequency-based threshold to trigger a BB alarm on stop B_j given BS^{b_j}
ρ	user-defined number of discrete bins employed to calculate ψ
b_ν	bus stop for which the BB event is predicted to occur
act^j	cor. action to be deployed once a BB alarm is triggered on b_j for the downstream stops
χ	symmetric user-defined min. threshold for the BB likelihood required to deploy a cor. action
χ_{BH}	min. threshold for the BB likelihood required to deploy Bus Holding
χ_{SS}	min. threshold for the BB likelihood required to deploy Stop Skipping
HT_k	Total Bus Holding Time to deploy to trip k
HT_k^i	Bus Holding Time to deploy to trip k on a given bus stop b_i
ζ_0, ζ	user-defined boundaries for the Total Bus Holding Time HT_k (in seconds)

Table 5.2: Notation and symbols about the Simulations and Passenger Demand Model employed along this Chapter.

RTV	Run Time Variation on a given route
SAT	Scheduled Arrival Time of a given trip
AAT	Actual Arrival Time of a given trip
AWT	Average Waiting Time for the passengers of the trips running a given route
$AIVT$	Average In-Vehicle Time for the passengers of the trips running a given route
B^k	total number of passenger boardings on a route during a given trip k
$PAV_{z,k}^{b_j}$	the arrival time of the passenger z to b_j immediately before trip's vehicle k arrival at b_j
$bs_{z,k}/as_{z,k}$	boarding/alighting stop of a given passenger z on trip k
bo_k^i/al_k^i	number of boardings/alightings of the trip k on the bus stop b_i
o_{max}	maximum passenger capacity of a given bus vehicle
o_k^i	occupancy of the bus k after the boardings/alightings at bus stop b_i
υ	user-defined frequency's percentage to be used on calculating the passenger arrivals
$de_{k,k+1}^i$	num. of passengers arrived to the stop b_i during the headway between k and $k+1$
$\lambda_{min}, \lambda_{max}$	user-defined minimum/maximum threshold for the value of $\lambda(k)$
df_i	descendant demand factor of the bus stop b_i
φ	expected percentage of the route completed by any passenger on a given trip
$ns_{z,k}$	number of stops traversed by a given passenger z during trip k
$fas(z,i,k)$	function that determines whether the passenger z alighted on the stop i during the trip k
$SIM2, SIM1$	simulations run by deploying/not deploying automatic corrective actions
$\Delta_{BH,g}^j$,$\Delta_{SS,g}^j$	variation on T_g^j provoked by deploying a cor. action (Bus Holding or Stop Skipping) on a previously departed trip k
ξ	user-defined constant boarding time per passenger
dwT_{min}, dwT_{max}	user-defined boundaries for the dwell time
Δbo_k^g	variation on the boardings of the trip k on a stop b_g imposed by a given corrective action
$\mathbb{T}_k^g$	arrival time of the trip k to the stop g of a given route affected by a corrective action

Eight of these routes are depicted on the road network in the urban area of Porto in Fig. 5.2. The orange dots represent the bus stops of each route.

5.1.1 Preprocessing

The origin of this data is the same than the one described in Section 4.1. Conversely to that dataset, this one is stop-based rather than trip-based. The data was sorted using the timestamps of vehicle's location associated to each link. The pairs were matched by identifying the records containing each trip's arrivals/departures with the defined schedule (which had the bus stops order and the scheduled arrival times to some of these stops - i.e. time point stops). Based on this information, it is possible to build route datasets. Each dataset has one entry for each trip containing the following information: starting date of the trip, bus vehicle model, Driver ID, day of the year, type of day (normal day, holiday and floating holidays), departure time from a given bus stop and a stop ID.

As part of the preprocessing phase, the raw route datasets were processed in order to make it suitable for later stages. The final route datasets have one entry for each stop visit along with the respective date (mapped as an incremental sequence starting in 1 for 01/01/2010), its type (weekdays - MON to SUN, and a day type - 1 for working days, 2-6 for other day types i.e.: holidays and strikes), a timestamp, the stop id and the link travel time from the previous stop.

Similarly to the dataset presented in Section 4.1, some data on link travel times is missing in the dataset. Not surprisingly, its percentage is larger than the one exhibited by the dataset used on the schedule coverage validation (roughly 10% of the total information - see Table 5.3). To overcome this issue on this

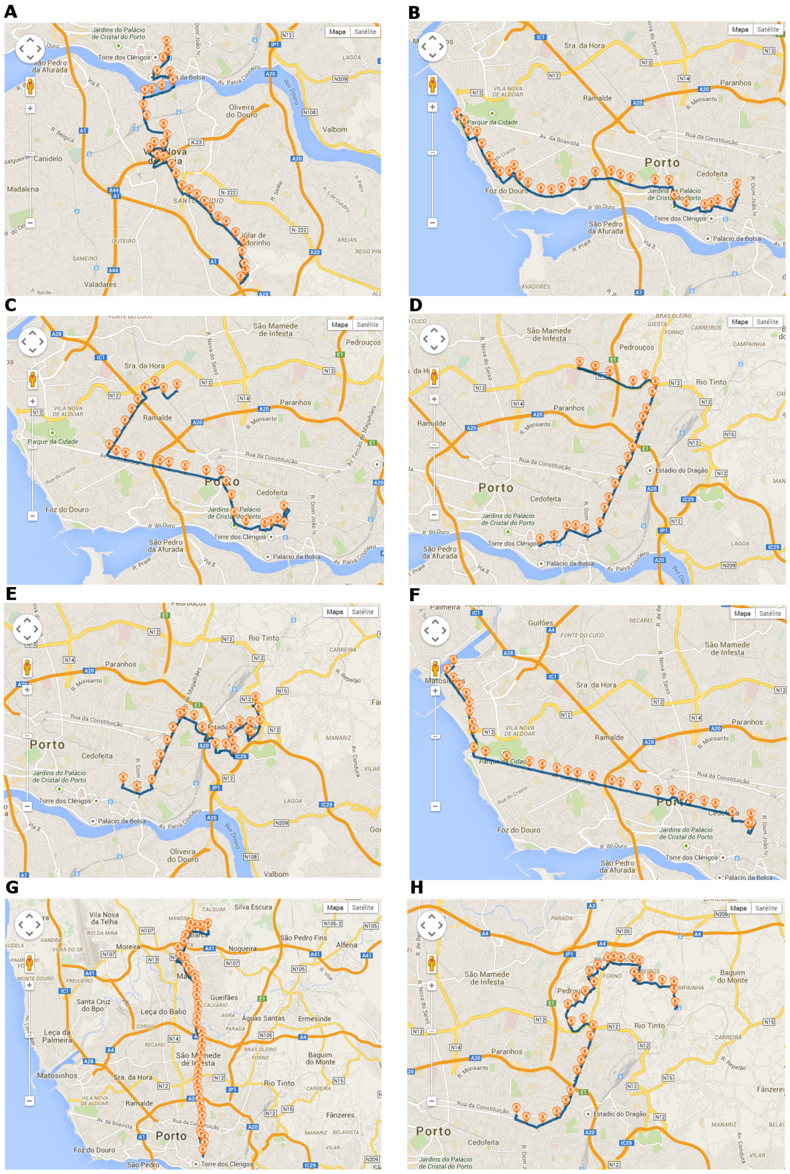


Figure 5.2: Illustration of some routes (one per line) considered over a geographical representation of the road network in Porto, Portugal. Image obtained from [STCP - Sociedade de Transportes Colectivos do Porto, 2013].

particular study, the links for which there is no information are not considered in our predictive framework[1].

[1] The predictions are done and updated as for any other links. However, as it is not possible to obtain the prediction residuals on those stops, they will not be considered to update the predictive model regarding the headway for the downstream stops. The reader can see Section 5.3 to know more about this particular issue.

Table 5.3: Descriptive statistics for each route considered. Headways in minutes.

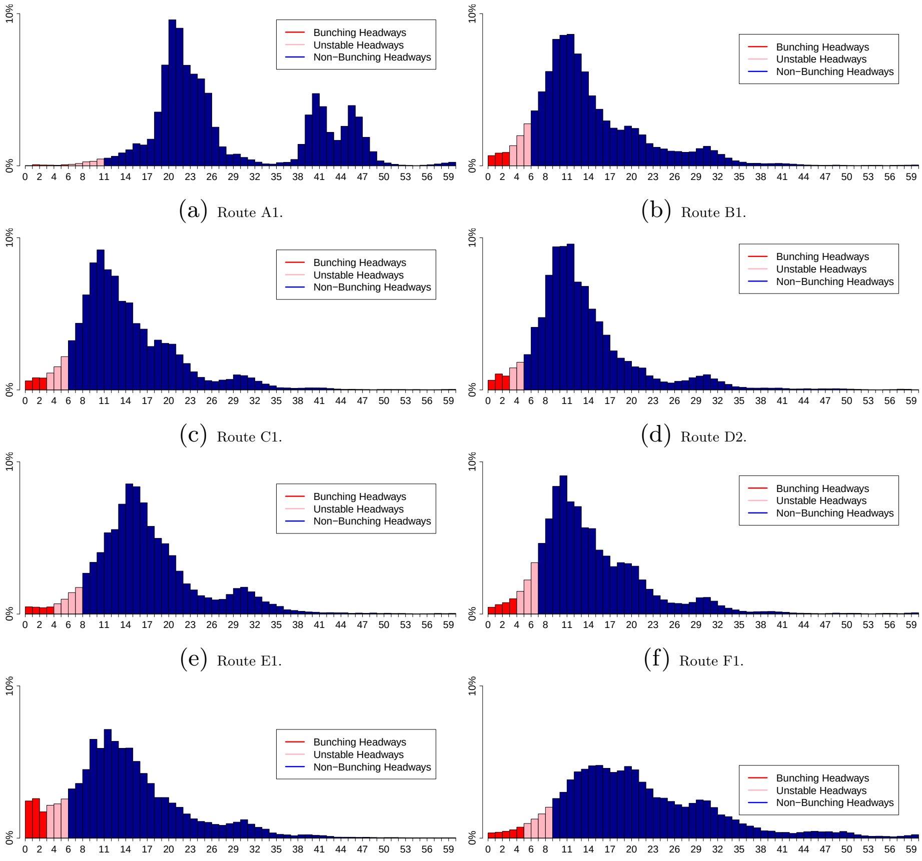
	A1	A2	B1	B2	C1	C2	D1	D2	E1	E2
Num. of Trips	10108	10224	24554	24388	20598	20750	25862	25674	18651	18940
Nr. of Stops	18	18	30	30	26	26	22	22	26	26
% of missing data	0.11%	0.09%	5.05%	4.93%	11.44%	6.70%	3.55%	1.20%	8.01%	4.98%
Min. Daily Trips	28	28	50	51	44	45	62	63	51	63
Max. Daily Trips	45	44	98	97	76	76	95	91	67	91
Min. Headway	20	18	10	9	10	11	9	9	14	14
Max. Headway	120	120	61	66	100	92	62	72	111	111
Avg. PH Headway	28.53	30.99	19.81	19.92	16.01	15.64	14.91	14.49	21.35	16.52
Trips w/ BB	19	43	734	811	682	553	1009	885	291	211
% of Bunching	0.18%	0.42%	2.99%	3.33%	3.31%	2.66%	3.90%	3.45%	1.56%	1.11%
BB Avg. Pos.	53.78%	82.94%	58.41%	76.28%	63.22%	74.87%	60.19%	62.13%	53.51%	67.89%

	F1	F2	G1	G2	H1	H2	I1	I2
Num. of Trips	20054	19361	26739	26007	11319	11864	15691	14901
Nr. of Stops	32	32	45	45	31	31	24	24
% of missing data	4.34%	2.17%	10.74%	7.5%	0.25%	0.47%	2.25%	7.23%
Min. Daily Trips	56	57	65	71	29	29	35	35
Max. Daily Trips	85	84	100	101	39	42	59	54
Min. Headway	12	13	10	10	20	19	17	19
Max. Headway	112	120	60	101	120	120	120	120
Avg. PH Headway	24.31	24.81	14.44	13.92	31.01	30.65	23.82	22.15
Trips w/ BB	437	364	1917	1702	17	23	388	225
% of Bunching	2.18%	1.88%	7.17%	6.54%	0.15%	0.19%	2.47%	1.51%
BB Avg. Pos.	58.32%	68.55%	49.71%	53.63%	56.57%	52.75%	60.79%	69.70

5.1.2 Dataset Statistics

Table 5.3 presents summary statistics of the dataset. The columns correspond to each of the routes included in the case study. The table rows show the following values per route: (1) total number of trips; (2) number of stops; (3) percentage of missing data for link travel times; (4,5) the maximum/minimum number of trips per day; (6,7) the maximum/minimum planned headway; (8) averaged measured headway on Peak Hours (PH); (9) total number of trips which experienced a headway shorter than 25% of the planned headway at least once along their trip; (10) the average position where a BB took place along the route (e.g. 50% means that on average the BB events took place at a stop situated in the middle of the route). The headway distributions of eight routes are also presented in Figure 5.3.

It is possible to observe that the routes with the largest number of trips also exhibit the largest percentage of missing data (except for line D). The minimum planned headway among these routes is 9 minutes. Nevertheless, it is evident that headways vary considerably as could be observed in Figure 5.3. In particular, the thick left and right tails of the headway distributions, are especially pronounced for routes B1, C1, D2 and G1. These lines are characterized by short headways. Such tails illustrate that these routes often exhibit headways which are significantly shorter or longer than the scheduled ones. As a result, these routes exhibit the highest share of trips containing extremely short headways of less than 25% of the planned headway (between 3% and 7% of all stop-visits). While all routes are subject to headway variations, the extent of these variations vary among the case study lines due to differences in the underlying traffic conditions and demand profiles.



(a) Route A1. (b) Route B1.
(c) Route C1. (d) Route D2.
(e) Route E1. (f) Route F1.
(g) Route G1. (h) Route I1.

Figure 5.3: Sample-based Headway (discrete) Distribution for eight routes of this study during the peak hours(truncated between 0 and 1h). Times are in minutes.

5.2 Related Work

Previous studies have deployed a range of analytical and simulation models to represent the dynamics of the bus service operations and evaluate the impacts of alternative control strategies. Early analytical studies that have examined the BB phenomenon and the characteristics of its instability that could be triggered by recurrent perturbations include Newell and Potts [1964]; Chapman and Michel [1978]; Powell and Sheffi [1983]. The latter devised a probabilistic model which built a set of recursive relationships to calculate the p.d.f. to validate the hypothesis of forming a platoon of vehicles on each stop. More recently, Daganzo [2009] and Daganzo and Pilachowski [2011] developed analytical models to assess the impacts of an adaptive control strategy which adjusts bus dwell time at stops and the running times between successive stops based on the respective headways.

Transit control strategies consist of a wide variety of operational methods aimed to improve transit performance and level of service. Holding strategies are among the most widely used transit control methods aimed to improve service regularity [Abkowitz and Tozzi, 1987]. In order to design and implement corrective actions, both the location - where the control decisions should be deployed [Turnquist and Blume, 1980; Abkowitz and Engelstein, 1984; Eberlein *et al.*, 2001; Sun and Hickman, 2008; Cats *et al.*, 2014] and the how - the criteria for intervening and its specification [Fu and Yang, 2002; Koutsopoulos and Wang, 2007; Cats *et al.*, 2012] - must be determined. In [Delgado *et al.*, 2009], a global control unit optimizes the holding times by solving a deterministic rolling horizon mathematical programming model which minimizes total passengers' waiting times.

Most of the abovementioned State-of-the-Art on this topic departs from the assumption that the probability of BB events is minimized by maximizing headway stability. This is achieved by either minimizing the difference between the actual headway and the scheduled one or by minimizing the discrepancies between successive headways. Notwithstanding its validity, this approach requires multiple control actions (i.e. speed modification, bus holding, etc.) which may impose high mental workload for drivers and result with low compliance rates.

Hereby, we propose a **proactive** rather than a **reactive** operational control framework. The basic idea is to estimate the likelihood of a BB event occurring further downstream to then let an event detection threshold triggers the deployment of a corrective control strategy. This methodology is described along the next Section.

5.3 Methodology

The occurrence of a BB event is subject to stochastic processes and hence difficult to predict. Notwithstanding, current system states may uncover such future occurrences. To do so, it is not sufficient to mine historical AVL data as there is no obvious trend or a simple *static* association rule which can explain such events. Consequently, an *off-the-shelf* Machine Learning method will not be sufficient to handle this specific problem.

This Section describes the details of a stepwise learning methodology to detect and then prevent BB in real-time. It utilizes simultaneously historical and real-time AVL data. The framework works on two different parts: (I) **BB Event Detection** and (II) **Corrective Action Deployment**. The first part is an Advanced Machine Learning framework composed of the following three steps:

- (I-1) an offline regression method is used to predict the **Link Travel Times** (i.e. the time interval between the arrival times at two consecutive bus stops) for every trip in the following day (the forecasting horizon) using some of the most *recent* days (the learning period) to train our model;
- (I-2) these predictions are constantly refined (i.e. online learning) using (I-2a) trip-level information as well as (I-2b) stop-based information. Both steps are based on the **Perceptron's Delta Rule** [Rosenblatt, 1958]

by reusing each prediction's residuals to improve successive predictions. After two consecutive trips of interest depart from their origin stop, a BB monitoring framework is triggered;

- (I-3) this framework estimates a **likelihood** of a BB event to occur at downstream stops by assuming that headway is normally distributed.

Given a certain user-defined threshold, a BB detection alarm is *launched*. The second part consists of the following two steps:

- (II-4) selecting one out of two possible **corrective actions** (Bus Holding or Stop Skipping) based on the relative headway deviation;

- (II-5) finally, the exact details of action implementation are specified based on service conditions and the designed set of corrective actions.

Parts I and II of the methodological framework are illustrated in Figures 5.4 and 5.5, respectively.

5.3.1 Step I-1: Link Travel Time Prediction

Let the trip i of a given bus route be defined by $T_i = \{T_i^1, T_i^2, ..., T_i^s\}$ where T_i^j stands for the arrival time of trip i at bus stop j and s denotes the number of bus stops along the route.

Consequently, the *observed* headways between two buses at stop j running on consecutive trips k,$k+1$ is defined as follows

$$H_{k,k+1} = \{H_{k,k+1}^1, H_{k,k+1}^2, ..., H_{k,k+1}^s\} : H_{k,k+1}^j = T_{k+1}^j - T_k^j \tag{5.1}$$

Under optimal conditions, the headway between two consecutive trips is expected to be *constant* (i.e. $H_{k,k+1}^i \simeq f_{k,k+1}, \forall i, k$).

However, in reality bus services are subject to uncertainty that results with service variability as discussed above. A BB event is expected to occur when headways become **unstable** until eventually forming a platoon. The operational control framework proposed in this thesis calls for the deployment of corrective actions when the headway deviates beyond a certain threshold. The corrective action is designed to recover to acceptable headway levels. The threshold that activates an intervention could be defined by the operator. The BB occurrence is hence defined as a boolean variable as follows

$$\text{BUNCHING} = \begin{cases} 1 & \textbf{if } \exists\, H_{k,k+1}^i \in H_{k,k+1} : H_{k,k+1}^i < \eta \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

where η stands for the headway-based minimum threshold to consider a BB event between two consecutive bus trips k and $k+1$. The definition of the η value is usually made as a function of the planned headway, i.e. $f_{k,k+1}$.

Consequently, it is possible to devise a *recursive* relationship between BB occurrences, the observed headway and the arrival time of a given trip i to a bus stop of interest l, i.e. T_i^l. Let the arrival time be defined as follows

$$T_i^{l+1} = T_i^l + dwT_i^l + RT_i^{(l,l+1)} \tag{5.3}$$

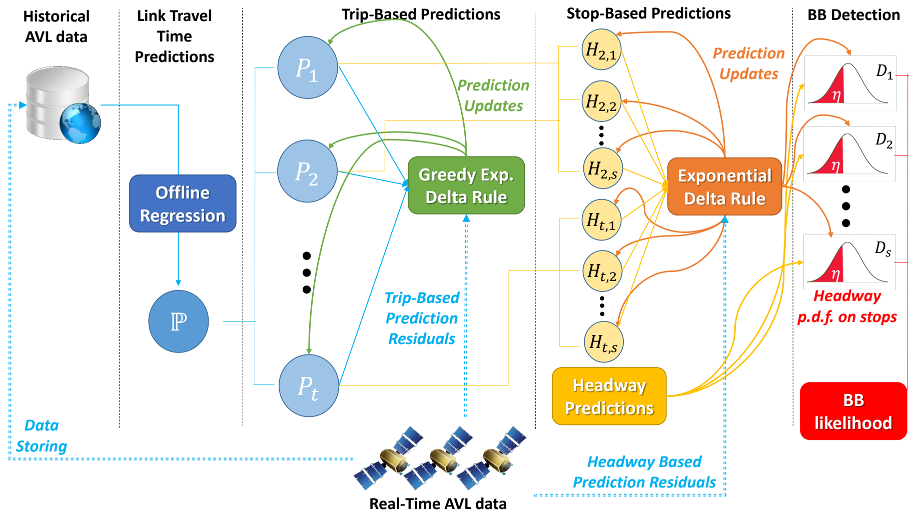


Figure 5.4: Illustration on the BB detection framework (part I).

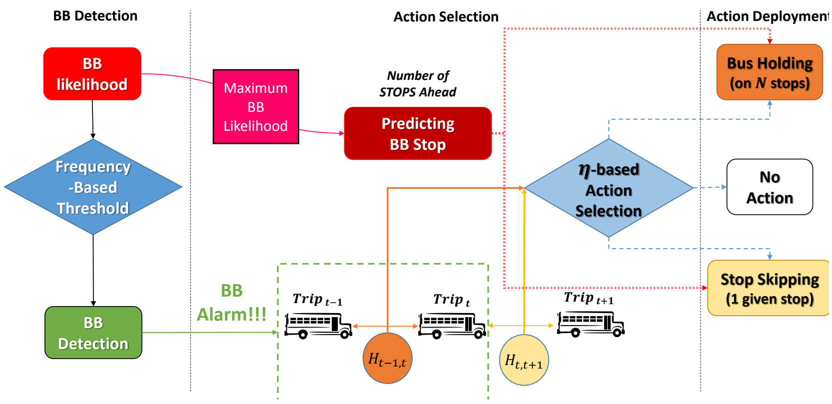


Figure 5.5: Illustration on the BB detection framework (part II).

The Link Travel Time (LTT) between two stops can be defined as follows:

$$LTT_i^{(l,l+1)} = dwT_i^l + RT_i^{(l,l+1)} \tag{5.4}$$

By observing the eqs. (5.1,5.3,5.4), it is possible to infer the following recursive relationship between headways measured on consecutive bus stops

$$H_{k,k+1}^{l+1} = H_{k,k+1}^l + LTT_{k+1}^{(l,l+1)} - LTT_k^{(l,l+1)} \tag{5.5}$$

Logically, it is possible to infer the future values of $H_{k,k+1}$ based on the predictions on the future values of LTT. Hereby, we propose to perform **long-term TTP** based on the dataset described in Section 5.1 in order to approximate the headways between every pair of consecutive trips on a daily basis. Departing from the work of [Mendes-Moreira *et al.*, 2012], it is possible to formulate the Link Travel Time prediction problem as an inductive learning regression problem. It involves inferring the following function

$$\bar{f} : X \to \mathbb{R} : \bar{f}(x) = f(x), \forall x \in X \tag{5.6}$$

where X stands for the *feature* set (i.e. a set of explanatory variables from which it is possible to establish dependences with the LTT; e.g. time of the day, type of day, etc.) and f represents the unknown explanatory function. The algorithm used to obtain the $\bar{f}$ is denominated *learner*. This type of algorithms usually scan one or multiple times a given training set where the true values of $f(x)$ are known to **generalize** a function that is able to output such values for *unseen* samples (i.e. future values of LTT).

Following [Mendes-Moreira *et al.*, 2012], we use a dynamic training set by employing a **sliding window** which only considers the most recent data (i.e. LTT for the latest θ days, where θ is an user-defined parameter) to train a $\bar{f}$ able to predict the LTT values for all the trips that take place on the following day.

By doing so, we aim to obtain an optimal fit of $\bar{f}(x)$ for a given training set (i.e. $\bar{f}(x) = f(x)$). This type of learning tasks is often denominated offline learning (as defined in the beginning of Chapter 3). They aim to find an explanatory function that performs an *optimal* fitting of unlabelled new data based on a given training set. Such convergence to *optimality* is one of the key characteristics of this type of models. However, it is also regarded as its major drawback because it is unable to adjust itself to changes introduced in the process by stochastic events as discussed in Section 2.1.3, such as traffic jams, abrupt weather changes or unusual demand peaks.

With this aim, we propose a **hybrid learning** model - which combines offline learning and online learning models. The offline regression produces a *context free* prediction for the LTT distribution throughout a day on a given route while the online learning handles the constant drifts that the learning process introduces due to multiple stochastic events that arise during system operations. Such online learning task involves updating these predictions based on the residuals (i.e. the difference between the predicted and the actual values) produced by earlier predictions. The residual-based update procedure is described in the following section.

5.3.2 Step I-2: Delta Rule as a Residual-Based Update

One of the most well-known offline learning techniques for regression are Artificial Neural Networks (ANN) [Bishop and others, 1995]. ANNs are computational models inspired by the neuron's brain structure. Perceptrons are the basic component of an ANN [Rosenblatt, 1958]. They play the role of a neuron in human brain. Typically, a Perceptron receives a set of inputs and computes their weighted sum. The output of the Perceptron is computed by an activation function, e.g. sigmoid, of the weighted sum of the inputs. Learning in a Perceptron consists of finding the weights, typically using gradient descent, that minimize the squared difference between the outputs and real values. In ANNs, Perceptrons are organized in layers, where the output of one layer act as input to the next layer. ANNs have been used in TTP, for example in Chien *et al.*

[2002]; Chen *et al.* [2004]; Jeong and Rilett [2005]; Mazloumi *et al.* [2011].

The most typical type of ANNs is a **Feedforward Neural Network (FNN)** - where the information just moves forward, from the input nodes to the output nodes. One of the most well-known algorithms to train FNNs is the Backpropagation Algorithm [McClelland *et al.*, 1986]. It basically progresses the outputs of their nodes *forward* while the residuals are propagated *backwards* to update the network weights until a convergence criteria is met (e.g. the average residual is below a given threshold). Such learning task is performed by employing the **Delta Rule (DR)** [Stone, 1986]. Let w_{ji} be the weight of the link connecting the i_{th} input node (with an input value of x_i) to the j_{th} output node where y_j is the node's current output and t_j is the target output. The delta rule updates the weight by adding to the previous weight w_{ji} a given Δ_{ji} as follows

$$w'_{ji} = w_{ji} + \Delta_{ji}; \Delta_{ji} = \alpha(t_j - y_j)x_i \tag{5.7}$$

where α stands for an user-defined parameter (i.e. typically, $1 >> \alpha > 0$). By running through multiple iterations, this algorithm will *force* the weights to *converge* in order to find a *local minimum* of a function (i.e. to minimize the $\bar{f}(x) - f(x)$). Obviously, a reasonable knowledge of the problem is normally required to perform an adequate feature selection and parameter setting such as the number of hidden layers and learning rate α in order to successfully apply ANNs to TTP problems [Bin *et al.*, 2006].

Moreover, ANNs also require a comprehensive amount of data and computational power to allow the learning model **absorve** all the dependences between the input and the output values. Nevertheless, they are not able to adapt themselves to handle *unseen* concepts and drift their outputs accordingly. However, when we are facing a large-scale data stream of information - such as the LTT dataset communicated by each vehicle - we face an high latency source of spatiotemporal data. Is it possible to turn this high information latency into an *advantage* by employing an ANN-based learning?

The need to converge for a local minimum for the residual's values is the key for a wide range of applications of ANN in many different research fields. However, it is also its main limitation as its *ambition* to generalize **all** the dependences decreases its applicability to many cases where there is *no time* to carry out such a complex optimization process [2] Instead, we adapt the delta rule to **incrementally update** the predictions firstly obtained by the offline regression learning process. So, we propose to modify eq. 5.7 as follows

$$y'_i = y_i \times \Delta_{y_i}; \Delta_{y_i} = \alpha(t_{i-1} - y_{i-1}) \tag{5.8}$$

turning it into a first-order update rule where the next prediction y_i is updated as soon as the real value of the previous one (i.e. t_{i-1}) is known. It thus consists of adding a percentage of the residual of y_{i-1}, i.e. $r_{y_{i-1}} = (t_{i-1} - y_{i-1})$. The basic idea is that the learning model will not change dramatically within several hours

[2] to learn adequately the *concept drifts* in the data, a typical backpropagation algorithm must carry out the full optimization process including the most recent samples on its training set. However, such process may require a considerable amount of time - especially in complex problems like our own. Such amount of time may result on an optimal but deprecated target function - as the concept may have drifted again due to the most recent samples arrived in the meanwhile.

(i.e. one day) but it will instead drift gradually as a response to changes in the expected values (i.e. a large-scale traffic jam). As these types of phenomena are also temporary they also need to be forgotten as soon as they have terminated (i.e. as a response to a progressive decrease of the $r_{y\,i-1}$ value). We denominate this update Linear Delta Rule. This algorithm was successfully deployed for other online learning problems (see, for instance, the incremental computation of the ARIMA's weights proposed in Section 6.3.2).

To improve the model ability to **react**, the learning rate α may also be updated based on the residuals' progression (i.e. $\alpha(r_y)$). Such update can be computed as follows

$$\alpha(r_y)' = \alpha(r_y) \times (1 + \vartheta \times (1 + \kappa^2)) \tag{5.9}$$

where κ^2 sets the progression rate of $\alpha(r_y)$ and ϑ stands for the number of *consecutive* residual's with the same signal (i.e. positive/negative). Consequently, κ^2 is a *quadratic* learning rate (i.e. user-defined) that sets the rate on how the original learning rate α is updated while ϑ is a variable that refers to trending in our prediction (i.e. over/under estimation). For a given prediction i, it is computed recursively as follows

$$\vartheta_i = \begin{cases} \vartheta_{i-1} + 1 & \textbf{if } \frac{r_{y\,i-1}}{r_{y\,i}} = 1 \\ 0 & \text{otherwise.} \end{cases} \tag{5.10}$$

where $\vartheta_0 = 0$. The variation of delta rule described in eqs. (5.9,5.10) is denominated as **Exponential Delta Rule** (Exponential DR). This algorithm was also successfully deployed for other online learning problems (see, for instance, the incremental exponential adaptation of the interval sizes in [Nunes *et al.*, 2012]).

The Exponential DR is applicable whenever a more sensitive reaction to changes on the residuals is desirable as compared with the Linear DR which it is more generic first-order update rule. Sometimes, the Exponential DR can also be applied directly to the output value as the learning rate α is also learned from the residual's distribution (i.e. $\alpha(r_y)$). Consequently, it is possible to re-write eq. 5.9 as follows

$$\Delta_{y_i} = \alpha(r_y) \times y_{i-1} \tag{5.11}$$

which gives an even greater reactivity to the learning model [Nunes *et al.*, 2012]. By doing so, this model is named as **Greedy Exponential DR**.

This type of updating rules can be considered as time-evolving *hidden layers* which aim to approximate concepts which were unknown on the generalization made by the offline regression learning process. In the following couple of subsections, we detail the application of these rules to incrementally update the LTT predictions.

5.3.3 Step I-2a: Trip-Based Link Travel Time Update

Let e denote the last trip completed before the current trip c starts. The trip-based refinement compares the predictions of e, i.e. $P_e = \{P_e^1, P_e^2, ..., P_e^s\}$ with their real values T_e in order to update the value of P_c. Firstly, we compute the residuals as $r_e = T_e - P_e$ and then its average value (i.e. r_P) as follows

$$r_P = \mu_e = \sum_{i=1}^{s} \frac{{r_e}^i}{s} \tag{5.12}$$

Secondly, an user-defined percent-wise maximum threshold $0 < \phi << 1$ is employed to identify trips with an error larger than expected as $\phi \times f_{c,e}$. Then, a Greedy Exponential DR is employed to update the LTT predictions for the next trip P_c as follows

$$P'_c = P_c + \Delta_{P_c}; \Delta_{P_c} = (\alpha(r_P) \times P_c) \tag{5.13}$$

where the dynamic learning rate $\alpha(r_P)$ is updated as follows

$$\alpha(r_P)' = \alpha(r_P) \times \left(1 + \vartheta \times (1 + \beta^2)\right) \tag{5.14}$$

where β^2 stands for the user-defined *quadratic* learning rate of the trip-based $\alpha(r_P)$. The threshold $\phi \times f_{c,e}$ is employed over the learning rate of $\alpha(r_P)$ by constraining the progression rate of $\alpha(r_P)$ defined in eq. 5.10 as follows

$$\vartheta_c = \begin{cases} \vartheta_{c-1} + 1 & \textbf{if } \frac{\mu_e}{\mu_c} = 1 \wedge \mu_e > (\phi \times f_{c,e}) \\ 0 & \text{otherwise.} \end{cases} \tag{5.15}$$

These updates are performed **incrementally** (i.e. every time a link is traversed and the respective travel time becomes available). Note that the residuals are always calculated over the regression results P_c and not over the updated arrays P'_c. Hence, its computation is iterative but not recursive.

5.3.4 Step I-2b: Stop-Based Headway Update

Given the updated predictions of two consecutive trips P'_c, P'_{c+1}, it is possible to obtain the predicted headways $E_c = P'_c - P'_{c+1}$ while the actual headways are computed as $H_c = T_c - T_{c+1}$. The calculus of E_c works as an offline prediction as it does not use information about the current headway. The second refinement uses the headway residuals $HR_c = H_c - E_c$ to update E_c stop-by-stop. Incrementally, we obtain online headway predictions as $\mathbb{E}_c^i = H_c^{i-1} + E_c^i - E_c^{i-1}, \forall i \in \{2, s\}$. The problem is to update the headway online prediction for the next stop (i.e. E_c^i) given the value of HR_c^{i-1}. To this end, we employ the Exponential DR, implemented with the following first-order rule:

$$\mathbb{E}_c^i = \mathbb{E}_c^i + \Delta_{\mathbb{E}_c^i}; \Delta_{\mathbb{E}_c^i} = \gamma^i(HR_c, H'R_c) \times HR_c^{i-1} \tag{5.16}$$

The dynamic learning rate $\gamma^i(HR_c, H'R_c)$ is updated as follows

$$\gamma^i(HR_c, H'R_c)' = \begin{cases} \gamma^{i-1} \times (1 - \gamma^{i-1}) \text{ if } |HR_c^{i-1}| > |H'R_c^{i-1}|, \\ \gamma^{i-1} \times (1 + \gamma^{i-1}) \text{ otherwise.} \end{cases}$$
$$\text{subject to } \gamma(HR_c, H'R_c^{i-1}) \in [\gamma_{min}, \gamma_{max}] \tag{5.17}$$

where $|HR_c|$ stands for the absolute residuals for the Headway's offline prediction E_c, $|H'R_c|$ denotes the absolute residuals for the Headway's online prediction $\mathbb{E}_c$ and $[\gamma_{min}, \gamma_{max}]$ stands for an user-defined boundary for the γ_r's range of values. Again, the entire headway array $\mathbb{E}_c^q, q \in \{i+1, s\}$ is constantly updated with the most recent headway value H_c^i as soon as it becomes available. This scheme provides a certain flexibility to handle the real-time stochasticity associated with headways. By performing these two steps, it is possible to maintain distinct levels of information to approximate the real-time link travel times incrementally. The propagation of our updates to downstream stops along the trip is key to BB anticipation, as explained in the following section.

5.3.5 Step I-3: BB Event Detection

A probabilistic framework for detecting a BB event at downstream stops is proposed. The **likelihood** of a BB event to occur at any of the downstream stops between two consecutive trips $(k, k+1)$ is computed by inferring the short-term probability distribution function (p.d.f.) of their headways $H_{k,k+1}$ at each stop, i.e $D(H^{b_i}_{k,k+1}) \equiv D^{b_i}, \forall i \in \{1..s\}$.

Let $M^{b_i}_k = \{M^{b_i}_{k-1}, ..., M^{b_i}_{k-\tau}\} : M^{b_i}_j = \left|H^{b_i}_{c,c+1} - \mathbb{E}^{b_i}_{c,c+1}\right|$ denote an array containing the most recent τ residuals of the headway predictions made at bus stop b_i, where τ is an user-defined parameter that defines the short-term memory size. To calculate such a p.d.f., it is postulated that

Assumption 5.1. *The Headway p.d.f. on a bus stop b_i, i.e. D^{b_i}, follows a* ***Gaussian distribution*** *defined as $D^{b_i} \sim \mathcal{N}(\mu_{b_i}, \sigma_{b_i})$.*

where μ_{b_i} is the expected headway value defined by $\mu_{b_i} = \mathbb{E}_{c,i}$ while σ_{b_i} is approximated by computing the median value of the recent prediction residuals (i.e. $\tilde{M}^{b_i}$).

Considering the hypothesis of a BB event occurring at this specific stop, we can express its likelihood as $p(BB^i_{k,k+1}) = p(H^i_{k,k+1} \leq \eta \mid \mathbb{E}^i_c, M^{b_i})$. This definition allows to quantify the p-value of a BB event to occur at a certain stop. It is then possible to quantify a Bunching likelihood for all downstream stops (and also to update them each time we obtain a more up-to-date headway value).

The aforementioned assumption 5.1 and the approximations made on the calculus of its parameters might induce a certain error because the headway distribution on a given stop may follow, on some circumstances, a different type of p.d.f. [Gentile *et al.*, 2005]. For simplicity and to allow its incremental computation, we have considered they all to follow a Gaussian distribution. To handle the error introduced by such assumption, a Bunching Score (BS^{b_j}) is estimated for a given bus stop b_j based on the estimations of the headway distributions D^{b_i} between the trips c and $c+1$ for downstream stops. Let $\mathbb{D}^{b_j} = \bigcup_{i=j+1}^{s} p(BB^i_{k,k+1})$ be an ordered vector (descendant) containing the likelihoods for the downstream bus stops. The BS can be obtained as follows:

$$BS^{b_j} = \frac{1}{n^j} \sum_{i=1}^{n^j} \mathbb{D}^{b_j}; n^j = \lceil 3 - ((j-1) \times 3/s) \rceil : n^j \in \mathbb{N} \qquad (5.18)$$

where n^j is the number of likelihoods used to compute BS^{b_j}. n^j works as minimum agreement threshold to set how many stops should somehow *agree* on raising a positive alarm on BB. Finally, a BB is said to be likely to occur at stops downstream of b_j if it exceeds a threshold value, ψ, defined as follows

$$\psi(f_{c,c+1}) = 0.3 + [(f_{c,c+1} \mod \rho) * 0.1] : 0 < \psi(f_{c,c+1}) \leq 1 \qquad (5.19)$$

where ρ stands for an user-defined parameter for the number of threshold bins that should be employed.

By employing n^j when computing BS^{b_j}, the method aims some sort of *consensus* by requiring high BB likelihoods for multiple bus stops whenever the BB events are being predicted at an upstream segment of a given route (i.e. longer forecasting horizons). After a BB alarm is triggered on a given bus stop b_j, a corrective action is implemented. In the next section, a process to automatically determine which action should be deployed in each situation is described.

5.3.6 Step II-4: Selecting a Corrective Action

In this study, two corrective actions are considered: (1) Bus Holding and (2) Stop-Skipping. The reason to do it so is their simplicity, easy communication and deployment [Delgado *et al.*, 2012]. Once a BB event between two consecutive trips $c, c+1$ is predicted for stops located downstream of b_j, it is essential to determine which control strategy should be deployed.

The procedure for strategy selection starts by determining which is the bus stop where the BB event will most likely occur, i.e. b_ν - which is determined by selecting the stop that maximizes the BB likelihood, as described in the following equation:

$$b_\nu = \underset{b_i}{arg\,max}\ p(BB^i_{c,c+1}) : j < i \leq s \tag{5.20}$$

Let $act^j \in \{0, 1, 2\}$ be the corrective action to be applied given that a BB alarm is triggered for bus stop b_j. The value 1 corresponds to deploying *Bus Holding*, 2 implies *Stop Skipping* and 0 does not involve any intervention. The value of act^j is selected based on the headways between the current trip c, the previous one $c-1$ and the following one $c+1$, where act^j is determined as follows[3].

$$act^j = \begin{cases} 2 & \textbf{if } \chi_{BH} \leq p\left(H^{b_\nu}_{c,c+1} \leq \eta\right) \geq p\left(H^{b_\nu}_{c-1,c} \geq ((2 \times \sigma_{b_\nu}) - \eta)\right), \\ 1 & \textbf{if } p\left(H^{b_\nu}_{c,c+1} \leq \eta\right) < p\left(H^{b_\nu}_{c-1,c} \geq ((2 \times \sigma_{b_\nu}) - \eta)\right) \geq \chi_{SS}, \\ 0 & \text{otherwise} \end{cases} \tag{5.21}$$

where χ_{BH} and χ_{SS} stands for two user-defined minimum thresholds for the BB likelihood required to deploy an action for a given PT system. The following step prescribes how the chosen strategy is implemented. The idea behind eq. 5.21 is to deploy stop skipping to address very particular situations, when we need not only to correct the headway between the current vehicle and its subsequent one, but also the one between the current vehicle and the following one. Such need arises on situations where this last pair is experiencing very long headways, as illustrated on Figure 5.6.

5.3.7 Step II-5: Implementing a Corrective Action

Once selected, the implementation of the control strategy has to be specified. If $act^j = 1$, then the holding time for bus $c+1$ is set as follows

$$HT_{c+1} = \eta - H^j_{c+1,c} + 10 : HT_{c+1} \in (\zeta_0 \times \{1, .., \zeta\}) \tag{5.22}$$

where $\{\zeta_0, \zeta\}$ are user-defined boundaries for the Total Holding Time (in seconds) to realistically reflect the driver-communication system limitations [Cats

[3] note that the distributions' parameter values were omitted to simplify the equation's readability;

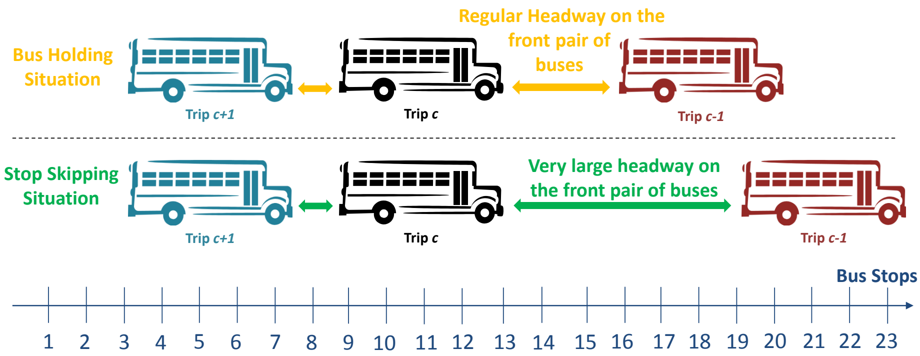


Figure 5.6: An illustration of correction action suitability - holding trip $c+1$ at the next stop (top), trip c skipping the next stop (bottom). Trip $c/c+1$ has to be slowed down or speeded up depending on subsequent headways.

et al., 2012]. Furthermore, an upper limit is set to holding time per stop due to passengers' acceptability reasons. The Total Holding Time is therefore **distributed** over multiple stops. Therefore, the bus holding time at each stop, HT_k^i, is computed as follows

$$HT_{c+1} = \sum_{i=j+1}^{\nu} HT_{c+1}^i : HT_{c+1}^i \in (\zeta_0 \times \{1,..,\zeta\}) \wedge \left(HT_{c+1}^i - HT_{c+1}^{i+1}\right) \leq \zeta_0 \tag{5.23}$$

If $act^j = 2$, the bus c is set to skip bus stop b_{j+1} or the subsequent stop if there is no possibility of informing passengers beforehand.

The feasibility of the abovementioned framework is investigated through computer-aided simulations which were executed using real-world data (described in Section 5.1). The details of these experiments along with their results are presented in the following Section.

5.4 Experiments

This Section begins by describing the experimental setup employed. Secondly, a tuning framework is proposed to adjust the parameter set of this framework for any case study of interest. Then, the evaluation metrics employed in this work are described, along with the details of the passenger demand profile generation process (which is unavailable for the present dataset). Finally, experiments' results are presented.

5.4.1 Experimental Setup

Throughout this work, the value η was set as $\eta = f_{k,k+1}/4$ following previous studies of this particular case study [Moreira-Matias *et al.*, 2012b]. For the offline regression problem, we also followed a simplified version of the experimental setup firstly proposed by Mendes-Moreira *et al.* [2012] using the Random Forest algorithm with a default parameter setting: {`mtry=3`, `ntrees=750`}. The main reason to do it so is that the accuracy of the initial offline regression stage is not our main concern. Consequently, we do not wanted to dispend such a high computational effort on determining the best regression algorithm for this particular dataset (as performed by Mendes-Moreira *et al.* [2012]). Instead, we proposed to use one (i.e. Random Forests) which is known by having a reasonable performance on this task [Mendes-Moreira *et al.*, 2012]. The value of θ was set to 1 (i.e. in weeks) following the same logic (in opposition to value 4 employed in [Mendes-Moreira *et al.*, 2012]). Even knowing the results reported in [Mendes-Moreira *et al.*, 2012], some prior experiments were made to validate the applicability of such experimental setup. To do it so, we tried to predict the round-trip times of two routes (i.e. line 205) using the last two months of the dataset described in Section 4.1 as testing set. The obtained results confirmed the existence of some convergence on the regression model obtained to the real output: the predictive model produced a Mean Absolute Error of $\simeq 150$ seconds against a value of 380 seconds obtained by a simple average of historical data on the same trip.

All the experiments were conducted using the R Software [R Core Team, 2012]. The proposed learning framework contains a total of eleven parameters: $\{\beta^2, \phi, \gamma^0$ (i.e. the first value of the learning rate)$, \gamma_{min}, \gamma_{max}, \tau, \rho, \chi_{BH}, \chi_{SS}, \zeta_0, \zeta\}$. A symmetric threshold was established in this case study for deploying corrective actions in this application (i.e. $\chi \equiv \chi_{BH} \equiv \chi_{SS}$). However, different values could be assigned to these parameters, depending on the operational plan devised for a given system [Carnaghi, 2014].

It is possible to divide this set into two types of parameters: prediction parameters (the first seven) and the deploying parameters (the last four). The deploying parameters must be set by the agency due to their close relationship with the Control policies already in place [Cats, 2014]. To this particular case, the deploying parameters were set to be $\{\chi = \chi_{BH} = \chi_{SS} = 0.5, \zeta_0 = 30$ (in seconds)$, \zeta = 4\}$. However, the prediction parameters require an adequate setting for minimizing the framework's error. Such tuning task is described below.

5.4.2 Parameter Tuning

The parameters $\{\beta^2, \phi, \gamma^0, \gamma_{min}, \gamma_{max}, \tau, \rho\}$ can also be divided into three different classes: 1) the update rule for the headway predictions (i.e. the first five), 2) the headway p.d.f. estimation (i.e. τ) and the likelihood threshold for event detection (i.e. ρ). The variation induced to this framework by changing one of these parameter values is not homogeneous (e.g. a variation on the learning rate β^2 will affect the output values more than a change in the value of τ). Following the previous experiments of the prediction task [Moreira-Matias *et al.*, 2014a], the value of γ^0 was set as $\gamma^0 = 0.1$ in a daily basis (as this value just affects the first stop-based update). The value of ρ was set to $\rho = 360$ seconds [Moreira-Matias *et al.*, 2014a].

The values of the remaining parameters must be tuned for each individual route since they relate to 1) the **stochasticity** of each BB process on a given route (i.e. $\{\phi, \gamma_{min}, \gamma_{max}\}$) and 2) the **reactivability** to sudden changes in such processes (i.e. $\{\beta^2, \tau\}$) - which is not necessarily correlated with the one exhibited by the remaining routes.

Table 5.4: Resulting Parameter Setting.

Route	ϕ	γ_{min}	γ_{max}	β^2	τ	**Route**	ϕ	γ_{min}	γ_{max}	β^2	τ
A1	0.100	0.010	0.300	0.010	10	**A2**	0.100	0.010	0.300	0.010	10
B1	0.050	0.010	0.400	0.300	5	**B2**	0.050	0.010	0.400	0.300	3
C1	0.050	0.005	0.300	0.300	5	**C2**	0.050	0.010	0.400	0.300	5
D1	0.075	0.010	0.400	0.500	3	**D2**	0.100	0.010	0.400	0.300	3
E1	0.050	0.005	0.300	0.300	5	**E2**	0.100	0.005	0.300	0.100	10
F1	0.050	0.005	0.300	0.300	5	**F2**	0.050	0.005	0.300	0.300	5
G1	0.050	0.005	0.300	0.300	5	**G2**	0.050	0.005	0.300	0.300	5
H1	0.050	0.005	0.300	0.100	5	**H2**	0.100	0.005	0.300	0.100	5
I1	0.050	0.005	0.300	0.300	5	**I2**	0.050	0.005	0.300	0.300	5

To carry out such tuning, we employ a simplified version of the Sequential Monte Carlo method [Cappé *et al.*, 2007]. It consists of *randomly* sampling data from the training set regarding subsets of its feature space (typically, 10% to 30%) to evaluate what is the combination of parameter values which performs globally better, on average for these samples. Ideally, the application of this framework to this problem would consist on randomly selecting a certain number of individual days of prior data (e.g. from the previous year) which could cover most of the possible cases (e.g. days from every months, every daytypes and weekdays). However, in our case, the first seven days of January of 2010 are the only true training set available - since the remaining days are already being used to validate our methodology, which does not contain a sufficient amount of data to carry out such analysis. To overcome this limitation, we overlapped a bit the test set by using the entire month of January to carry out the analysis. Ten days were then randomly selected containing every possible weekday and daytype. Even so, we do want to highlight that the abovementioned procedure should be ideally conducted on prior data to achieve a satisfactory tuning of the parameter set.

All possible combinations of the following parameter settings were considered in our experiments: $\phi = \{5\text{E}^{-3}, 1\text{E}^{-2}, 2\text{E}^{-2}, 5\text{E}^{-2}, 7.5\text{E}^{-3}, 0.1\}$, $(\gamma_{min}, \gamma_{max}) = \{(1\text{E}^{-3}, 0.3), (5\text{E}^{-3}, 0.3), (1\text{E}^{-2}, 0.3), (1\text{E}^{-3}, 0.4), (5\text{E}^{-3}, 0.4), (1\text{E}^{-2}, 0.4)\}$, $\beta^2 = \{1\text{E}^{-2}, 5\text{E}^{-2}, 1\text{E}^{-1}, 3\text{E}^{-1}, 5\text{E}^{-1}\}$ and $\tau = \{3, 5, 10, 15\}$. The obtained parameter setting is displayed in Table 5.4.

5.4.3 Evaluation Metrics

It is possible to divide the evaluation of our framework on three distinct dimensions: (i) the mean absolute error (MAE) (already defined on eq. 3.22) of the headway's predictions, (ii) the BB detection accuracy and (iii) the effect of the actions deployed in the PT network on travelers.

Concerning the first dimension, (i) a prequential evaluation (see Section 3.5) was performed by evaluating just the prediction made for the LTT prediction performed for the next bus stop.

In (ii) the Accuracy, the Precision and the Recall were employed to evaluate the BB event detection framework. A Weighted Accuracy was also employed to give greater weight to a false negative versus a false positive, in the detection of

BB events. It can be computed as

$$WAcc = \frac{(10 \times TP) + TN}{(10 \times TP) + TN + FP + (10 \times FN)} \quad (5.24)$$

The Average Number of Stops Ahead is also displayed to show the forecasting horizon that this framework can yield.

The most direct metric for dimension (iii) concerns the percentage of BB reduced by employing our automatic control actions. Such ratio can be computed as follows

$$BB_{reduction} = \frac{\texttt{BB\_Trips\_Without\_Actions} - \texttt{BB\_Trips\_With\_Actions}}{\texttt{BB\_Trips\_Without\_Actions}} \quad (5.25)$$

Nevertheless, the ultimate motivation to avoid the occurrence of BB events is to improve the global quality of the service provisioned. A BB event does decrease the passengers' perception of the service quality. Moreover, it also yields longer Passenger Waiting Times for passengers waiting at downstream stops. However, the deployment of corrective actions can potentially increase Passenger In-Vehicle Time due to prolonged times at stops and on-board inflicted by holding or stop skipping [Cats *et al.*, 2010]. It is therefore necessary to consider both the **Average In-Vehicle Time** (AIVT) and the **Average Waiting Time** (AWT) (already defined in eqs. (2.6,2.8) when evaluating alternative operational plans. By avoiding the occurrence of BB events, it is expected to reverse the well-known snowball effect of the BB process and hence reduce global AWT on a given route. The deployment should ensure that this is achieved without compromising global AIVT. In order to evaluate the success of such minimization task, two large set of simulations were performed: (SIM1) no actions were deployed on the route and (SIM2) actions were deployed accordingly to the framework described in Section 5.3.6.

In SIM2, two additional ghost trips were introduced whenever a BB alarm is triggered. This is done by re-running the two affected trips from the beginning applying the necessary corrective actions and not applying any action at all on the ghost trips. The idea is to estimate the variation on the dwell times experienced by both trips to predict the real impact on the vehicle's LTT and, consequently, on the AIVT and AWT. Such ghost trips also serve to evaluate the accuracy of our corrective actions in preventing the occurrence of such BB event (i.e. act_{ACC}).

In the absence of observations of passenger counting, a synthetic demand profile was devised based on the real world available LTT, on the network's schedule plan and on simple load assumptions, as presented in the following section.

5.4.4 Demand Profile Generation Procedure

Let o_{max} be the maximum passenger capacity of a bus running a given route. The occupancy of a given vehicle k after departing from a given bus stop i is given as follows

$$o_k^i = o_k^{i-1} + bo_k^i - al_k^i : o_k^i \leq o_{max} \quad (5.26)$$

where bo_k^i and al_k^i denote respectively the number of boarding and alighting passengers for trip k at bus stop i. In the absence of empirical data to calculate such values, the following assumptions were devised for constructing a demand profile:

Assumption 5.2. *On high frequent routes, it is assumed that the passengers arrival process follows an inhomogeneous Poisson process with a given rate* $\lambda(t)$.

It is known that the timetables are designed to accommodate variations in passenger demand levels by setting different service frequencies for different time periods and routes (Chapter 4 in [Ceder, 2007]). In order to introduce some perturbations in passenger demand, we sampled values from (λ) using a **Gaussian** p.d.f.. Such probability distribution is defined based on the frequency $f_{k,k+1}$ and some user-defined parameter $0 < \upsilon << 1$ which basically sets the amount of *white noise* introduced on our demand generation model. The sampling process is defined as follows

$$\lambda(k) \sim \mathcal{N}\left(\mu = \upsilon \times f_{k,k+1}, \sigma = \upsilon^3 \times f_{k,k+1}\right) : \lambda_{min} \leq \lambda(k) \leq \lambda_{max}, \lambda(k) \in \mathbb{N} \tag{5.27}$$

where $\lambda_{min}, \lambda_{max}$ are user defined boundaries for $\lambda(k)$. υ denotes the percentage of the frequency to be used when calculating passenger arrivals and $\lambda_{min}, \lambda_{max}$ are a minimum/maximum threshold for the value of $\lambda(k)$. Based on such *p.d.f.* definition, values for $\lambda(k)$ can be *sampled* for each trip k.

From empirical evidence, it is also known that passenger demand also varies along the route (e.g. [Munizaga and Palma, 2012]). This is captured by incorporating a linear descendant demand factor for each bus stop b_i, i.e. df_i. It can be computed as

$$df_i = \begin{cases} \frac{2\times(s-i+1)}{s} & \textbf{if } i \leq s, \\ 0 & \text{otherwise.} \end{cases} \tag{5.28}$$

Based on eqs. (5.27,5.28), it is possible to infer the calculus of bo_k^i as follows

$$de_{k-1,k}^i = \left(H_{k-1,k}^i \times \lambda(k) \times df_i\right) \tag{5.29}$$

$$bo_k^i = de_{k-1,k}^i + Nbo_{k-1}^i : bo_k^i \in \mathbb{N} \tag{5.30}$$

where Nbo_{k-1}^i stands for the number of passengers that were not allowed to board on the vehicle $k-1$ and $de_{k-1,k}^i$ is the number of passengers arrived to the stop b_i during the headway between k and $k-1$ (demand generated during the period of $H_{k-1,k}^i$). $Nbo_{k-1}^i > 0$ whenever the vehicle $k-1$ rides *full* after traversing stop i or if the stop i is *skipped* by bus k as a consequence of a corrective action.

It is assumed that passengers trip length varies between 25% and 50% of the respective bus route. The user-defined parameter $25\% \leq \varphi < 50\%$ is introduced where the number of stops traversed by a given passenger z to the trip k, i.e. $ns_{z,k}$ is assumed to follow an **lognormal** distribution, as defined on the equation below

$$ns_{z,k} \sim \ln\mathcal{N}\left(\mu = \ln\left(\varphi \times df_{bo_{z,k}} \times s\right), \sigma = \ln\left(\varphi^3 \times df_{bo_{z,k}} \times s\right)\right), \quad \text{subject to: } ns_{z,k} \geq 1, \mu > \sigma > 1, ns_{z,k} \in \mathbb{N} \tag{5.31}$$

where $bo_{z,k}$ represents the boarding stop of the passenger z on the trip k. Consequently, it is possible to obtain the alighting stop of z in k as

$$as_{z,k} = bo_{z,k} + ns_{z,k} : as_{z,k} < s \tag{5.32}$$

Like the boardings, the alightings also assume some realistic stochasticity by being sampled from a predefined distribution (and not from an exact mathematical definition). Again, the demand factor df_i is employed (i.e. the passengers boarded on the beginning of the routes are *likely* to traverse more stops than the ones boarded on its end). Let $fas(z,i,k)$ be a *boolean* function defined as follows

$$fas(z,i,k) = \begin{cases} 1 & \textbf{if } as_{z,k} = i, \\ 0 & \text{otherwise.} \end{cases} \tag{5.33}$$

The number of alightings of a vehicle/trip k at bus stop b_j can be hence computed as follows

$$al_k^i = \sum_{z=1}^{B^i} \sum_{j=1}^{i-1} fas(z,j,k) : al_k^s = o_k^{s-1} \tag{5.34}$$

Although being enough to compute $AIVT$, the definitions in eqs. (5.30,5.34) do not allow to compute the AWT as the arrival time of a given passenger z to a bus stop b_j, $PAV_{z,k}^{b_j}$ is unknown. To obtain such values, we reverse the effects imposed by the assumption 5.2 by inferring passengers arrival times, $PAV_{z,k}^{b_j}$, from the respective exponential distribution Exp $(\lambda(k))$. This is performed using the following steps: (a) $de_{k-1,k}^i + 1, \forall i,k$ values are sampled from Exp $(\lambda(k))$ to express the time between each passenger arrival; (b) the values are normalized to let their sum meet the total elapsed time $H_{k-1,k}$) by dividing each sampled value by their total sum and then multiplying all of them by $H_{k-1,k}$; (c) the arrival times are then incrementally summed to express the time elapsed from the departure of b_j to each passenger arrival time $PAV_{z,k}^{b_j}$ - which will force one of these values to be the total sum of values (i.e. $H_{k-1,k}$), and (d) the latter value is then removed to obtain the set of $PAV_{z,k}^{b_j}$ for the demand generated on b_j between the departures of $k-1$ and k, $de_(k-1,k)^i$.

The consequences of deploying a given corrective action on a given trip k also have to be captured in the simulation model by representing its effects on LTT. Let $\Delta_{BH,k}^j$ and $\Delta_{SS,k}^j$ stand for the change in arrival time of trip k to the following bus stops provoked by deploying a Bus Holding or Stop Skipping action at bus stop b_j, respectively. Such changes can be computed as follows

$$\mathbb{T}_k^g = T_k^g + \Delta_{BH,g}^j + \Delta_{SS,g} \tag{5.35}$$

$$\Delta_{BH,k}^g = \sum\nolimits_{i=j+1}^{g-1} HT_g^i : g > j \tag{5.36}$$

$$\Delta_{SS,k}^g = -dwT_k^j \tag{5.37}$$

$$dwT_k^j = de_{k-1,k} \times \xi + dwT_{min} : dwT_k^j \leq dwT_{max} \tag{5.38}$$

where dwT_{min}, dwT_{max} are two user-defined boundaries for the dwell time and $\xi > 0$ is a user-defined constant boarding time per passenger (which will correspond to an excess/reduction). Consequently, $\mathbb{T}_k^g$ denotes the LTT of k affected by the deployment of a corrective action on the network. However, by influencing headway stability, the characteristic recursive effect of the BB process (described in the introductory Section) is **reversed** (e.g. if some holding is imposed on k, bus $k+1$ will experience shorter dwell times as *some* of the demand will be accommodated by k). Such effects are also accounted on the simulation SIM2 by devising the following first order relationships

$$\Delta^{g}_{BH,k+1} = T^{g}_{k+1} - (\Delta bo^{g}_{k} \times \xi) + \Delta^{g-1}_{BH,k+1} \tag{5.39}$$

$$\Delta^{g}_{SS,k+1} = T^{g}_{k+1} - (\Delta bo^{g}_{k} \times \xi) + \Delta^{g-1}_{SS,k+1} \tag{5.40}$$

where Δbo^{g}_{k} stands by the change in the number of boarding passengers for trip k at bus stop b_g attributed to the corrective action deployment. Note that the recursive relationship imposed by eqs. (5.39,5.40]) is not necessarily constrained to the trip subsequent to the corrective action deployed $(k, k+1)$, but also to the subsequent ones $(> k+1)$ in a **snow ball effect**.

The parameters of this demand profile generation procedure were set to the following values: $\xi = 3$, $dwT_{min} = 10$, $dwT_{max} = 90$, $\upsilon = 0.2$, $\lambda_{min} = 60$,$\lambda_{max} = 180$,$\varphi = 0.25$ (all times in seconds). An illustrative example of the demand profile generated by this model is illustrated in Fig. 5.7. The results of the experimental simulations described above are presented in the following section.

5.4.5 Results

Table 5.5 contains the prediction results while the effects of the corrective actions are introduced in Table 5.6. The reader should analyse these Tables together with Table 5.3 to understand the differences among the routes.

5.5 Discussion

The performance of the headway predictive framework varies from route to route - even for opposite directions of the same line. It is peculiar to note, for instance, that the offline prediction performance on line G, is four times greater for one of the route-directions than for the other one. Notwithstanding, it is important to stress out that in both cases the predictive framework performs reasonably well (i.e. an error of $\simeq 30$ sec.). The online learning framework converged to the real output values by reducing the average error by more than 90% (i.e. with the exception of route I2).

The long BB prediction horizons (i.e. roughly 11 stops, on average), enables a gradual and incremental implementation of corrective actions. At the same

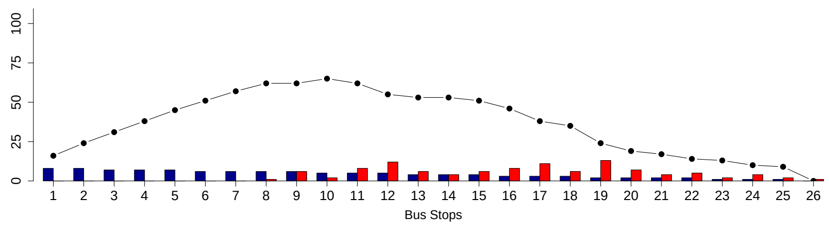


Figure 5.7: Demand Profile Generated for a given trip on the route C1. The red/blue bars represent the alightings/boardings on each stop, while the dashed line represents the bus occupancy's evolution throughout the trip.

time, the values of the precision are low - especially for certain routes (i.e. lines A and H) due to triggering more alarms than are really necessary (false positive). However, such pattern is admissible in the context of this problem - since it is desired to deploy corrective actions following a more **proactive** approach to the BB. The routes with lower precision values are those operated with lower frequencies. This suggests that the BB detection threshold values (i.e. χ and η) should not be uniformly specified for the entire network. This could be investigated in future studies. It is important to highlight that **more than 83% of the BB events** that prevailed in the case study dataset **were forecasted accurately**.

Table 5.6 summarizes the corrective actions implementation and their impact on passengers travel times. As expected, Holding is selected in most cases over Stop Skipping (i.e. 81.68%). However, it is important to highlight that in a substantial share of BB detection no action was taken (i.e. 30.66% for the route G2) which relates to the abovementioned need to set route-specific values of η (i.e. minimum headway threshold for BB). Noticeably, the applied framework did not prove effective for low-frequency routes (i.e. lines A and H). This is not surprising as the BB phenomenon is most prevalent on high-frequency routes and the corrective actions deployed in this study are designed to regulate headways on high-demand routes. Furthermore, note that the low value of χ (i.e. symmetric user-defined min. threshold for the BB likelihood required to deploy a cor. action), constrained the deployment of actions to a small subset of trips (i.e. between 3% and 7% of the total number of trips - check Table 5.3). and therefore yielded a reduction of only 4.46% when measured globally. This reflects however a substantial travel time savings given that a BB occurs. Moreover, the number of BB events have been reduced by 67.59% (out of the 83% of the BB forecasted). It clearly demonstrates the usefulness

Table 5.5: Experimental Results regarding the BB predictive framework. The **ALL** column corresponds for aggregated results (averaged for the prediction errors and summed for the total numbers of BB events). Times in Seconds.

	A1	**A2**	**B1**	**B2**	**C1**	**C2**	**D1**	**D2**	**E1**	**E2**
MAE offline regression	63.79	83.77	1671.54	765.33	1356.96	643.39	277.78	174.58	255.39	363.91
MAE inter-trip update	31.08	33.87	114.16	62.17	97.87	92.91	49.26	39.31	33.65	91.56
MAE incremental update	30.38	27.97	26.47	17.67	24.14	26.35	34.78	27.74	24.78	16.30
Accuracy	99.34%	99.45%	96.96%	97.86%	97.99%	96.34%	98.57%	98.24%	99.49%	99.24%
Weighted Accuracy	98.38%	97.13%	93.86%	91.81%	93.97%	93.57%	94.41%	93.06%	97.15%	97.46%
Precision	13.85%	36.73%	49.48%	65.97%	65.88%	40.85%	74.30%	72.51%	84.75%	62.09%
Recall	47.37%	41.86%	83.52%	73.61%	81.81%	83.18%	84.54%	78.08%	82.13%	81.57%
Avg. Nr. of Stops Ahead	5.42	3.44	13.18	15.99	11.85	14.78	9.02	8.91	10.21	10.67
Correct BB Predictions	9	18	613	597	558	460	853	691	239	172
Real BB Events	19	43	734	811	682	553	1009	885	291	211

	F1	**F2**	**G1**	**G2**	**H1**	**H2**	**I1**	**I2**	**ALL**
MAE offline regression	1475.72	1871.01	473.61	2776.57	1719.42	241.56	290.39	157.77	**814.91**
MAE trip-based update	124.99	148.85	40.65	123.76	105.88	34.40	39.42	31.76	**71.98**
MAE stop-based update	22.67	13.21	31.78	27.47	19.05	12.65	22.49	38.81	**24.71**
Accuracy	97.08%	97.83%	96.62%	93.83%	99.81%	99.76%	98.62%	98.44%	**98.06%**
Weighted Accuracy	94.56%	95.52%	95.72%	91.50%	99.19%	99.01%	94.70%	92.23%	**95.19%**
Precision	41.53%	45.70%	69.44%	51.67%	40.00%	42.42%	69.39%	48.33%	**54.16%**
Recall	83.07%	83.24%	94.47%	87.96%	58.82%	60.87%	78.87%	51.56%	**74.25%**
Avg. Nr. of Stops Ahead	13.88	15.08	12.96	14.51	11.81	6.05	13.90	11.96	**11.31**
Correct BB Predictions	363	303	1811	1497	10	14	306	116	**8630**
Real BB Events	437	364	1917	1702	17	23	388	225	**10311**

Table 5.6: Experimental Results regarding the deployment of the corrective actions. The **ALL** column corresponds to aggregated results (i.e. the sum of the cor. actions and the weighted average per routes using the number of trips). Times in Seconds.

	A1	**A2**	**B1**	**B2**	**C1**	**C2**	**D1**	**D2**	**E1**	**E2**
Total of Cor. Actions	57	43	1225	874	833	1101	1119	927	276	273
% Bus Holding	80.70%	88.37%	79.02%	89.25%	75.51%	95.00%	86.23%	85.01%	69.57%	86.08%
% Stop Skipping	8.77%	4.65%	9.06%	5.72%	15.61%	0.64%	5.99%	8.52%	25.36%	8.79%
% None	10.53%	6.98%	11.92%	5.03%	8.88%	4.36%	7.78%	6.47%	5.07%	5.13%
Avg. Total Holding Time	93.26	104.21	109.59	114.81	108.79	119.20	115.93	109.72	95.00	107.62
% of BB Reduction	50.98%	65.00%	66.08%	77.83%	65.48%	86.89%	72.09%	71.05%	61.83%	74.90%
AIVT w/ Actions	193	187	725	631	620	515	515	470	530	488
AIVT without Actions	193	187	732	632	614	508	502	463	525	488
AWT w/ Actions	1327	1345	872	1020	715	733	730	788	1288	1350
AWT without Actions	1327	1344	903	1034	749	751	800	839	1322	1380
% of Reduction of AWT	0.00%	0.00%	3.43%	1.35%	4.54%	2.40%	8.75%	6.08%	2.57%	2.17%

	F1	**F2**	**G1**	**G2**	**H1**	**H2**	**I1**	**I2**	**ALL**
Total of Cor. Actions	856	648	2554	2782	23	29	418	230	**14268**
% Bus Holding	83.18%	87.81%	89.78%	58.05%	73.91%	90.00%	72.48%	81.30%	**81.68%**
% Stop Skipping	10.40%	5.40%	4.42%	11.29%	21.74%	3.45%	25.12%	14.35%	**10.26%**
% None	6.43%	6.79%	5.79%	30.66%	4.35%	6.56%	2.40%	4.35%	**8.06%**
Avg. Total Holding Time	112.24	116.73	117.50	116.47	105.88	113.08	108.51	111.34	**111.16**
% of BB Reduction	68.66%	77.15%	66.91%	47.48%	63.64%	65.38%	59.07%	62.72%	**67.59%**
AIVT w/ Actions	685	928	601	597	739	586	698	498	**584.39**
AIVT without Actions	678	935	599	603	735	587	699	499	**560.41**
AWT w/ Actions	915	1070	801	888	1682	1720	1369	1501	**1038.40**
AWT without Actions	924	1082	933	983	1680	1725	1404	1547	**1043.65**
% of Reduction of AWT	0.97%	1.11%	14.15%	9.66%	-0.00%	0.00%	2.49%	2.97%	**4.46%**

of our framework on this particular application - which meets no parallel in the literature. It is also remarkable that such achievement does not induce an increase in the global In-Vehicle Time. Further gains could be obtained by setting optimal parameters for each route. A data driven approach to such problem could utilize the information about the actions currently deployed on each route.

5.5.1 Potential Deployment and Impact

The main prerequisites for the application of the proposed framework in real-world operational control of most major PT companies worldwide are already fulfilled: the existence of AVL data in real time, a control center that monitors these data also in real time and, finally, the means to establish communication between the control center and the drivers. The same cannot be said about the subjective conditions in terms of management perceptions. The prevailing attitude among PT companies is to regard BB as almost inevitable, considered to be a constant feature of bus service. Consequently, these companies do not assign the necessary means for its resolution. This is a question that has to be faced in the dialogue between researchers and companies, and the remaining of this Section is a preliminary essay to argue its importance.

The value of preventing BB is not limited to operational costs and the direct impact on passengers' experience in terms of travel times and crowding, but is also related to the overall perception of service quality. The implementation of an operational system such as the one proposed in this Chapter has therefore implications on the overall service perception. The feasibility of implementing such a system depends on the estimated operational costs that arise in case of BB. The following is an attempt to quantify these costs.

Each time a BB event occur, the bus driver and the vehicle of the subsequent trip may experience delays because the trip takes longer than planned. This delay is, at most, the frequency of the respective route, typically a short one; otherwise the occurrence of bus bunching would be unlikely. Since the occurrence of BB is caused by delays in the front bus and advances in the successive, the extra time $ET_c = f_{c,c+1} - H_c$ spent by the front bus c is in the interval $[0, ET_c]$. Assuming that no measures are taken in order to regulate the headway, it is expected that BB situations, once started in a route, will continue as long as the frequency remains high. Assuming the worst scenario $ET_c = f_{c,c+1}$, and assuming that there are u trips with the same frequency in that route since the beginning of bus bunching situations, there will be in the end $\frac{ET_c \times u}{2}$ extra time spent.

The cost of such situation is easy to calculate since there is an estimation of the cost per bus and per driver for each extra minute. This is of course, an upper bound for the real operational cost of bus bunching situations in terms of buses and drivers' duties. These operational costs could then be added to the benefits in terms of passengers travel time savings (based on the value-of-time estimations [Wardman, 2004]) to assess the overall benefits from implementing a framework for preventing BB. In addition to this tangible effects, the impact of BB on service image and hence attractiveness and ridership could be assessed using satisfaction surveys in order to support the decision making process.

5.6 Final Remarks

A novel real-time framework to prevent BB from occurring was proposed in this Chapter. It combines historical and real-time AVL data to predict the occurrence of BB events at downstream stops. The prediction output are then used to select and deploy automatic corrective actions. This framework consists of advanced Machine Learning methods which are able to gain foresight on the BB process. Experiments were conducted using a large-scale dataset of real world data collected in Porto, Portugal. The application yielded a reduction of 68% in the number of BB events. The results demonstrate that this framework can be readily deployed for mass transit systems across the world. Moreover, it is estimated that it could have a real impact on the passengers experience by decreasing the average expected waiting time on the stops by approximately 5% without causing an increase in in-vehicle times.

Future work includes carrying out experiments to optimally set the parameters η (i.e. an headway-based minimum threshold to consider a BB event) and χ (i.e. a minimum BB likelihood threshold to deploy a corrective action) for each individual route and possibly also time-dependent. Further research can use for this purpose data about the corrective actions deployed on each route. Moreover, the assumptions about the headway distribution (i.e. Gaussian) and their parameter calculus may be too far simplistic for some scenarios. Further research should be employed on considering more than one type of distributions.

The parameter estimation can also be improved by employing change detection techniques (e.g. the CUmulative SUM algorithm [Page, 1954]) able to avoid the inclusion of outliers on the calculus of the distribution's parameter - instead of using a simple median of the recent residuals).

The framework presented in this Chapter could ultimately be embedded into a decision support system that will be deployed in control rooms of PT agencies and operators.

Part III

Urban Mobility

Chapter 6

Short-Term Taxi-Passenger Demand Prediction

Nowadays, taxi plays a crucial role on the major urban areas worldwide. It represents a major contribution to improving the quality of the urban mobility by providing direct, fast and comfortable on-demand connections. The rising cost of fuel has been reducing the profit for both taxi companies and drivers. This leads to an unbalanced relationship between passenger demand and the number of running taxis, which in turn reduces the companies' profits and also the levels of passenger satisfaction [Schaller, 2007]. S. Wong presented a relevant mathematical model to express this need for an equilibrium in distinct contexts [Yang *et al.*, 2001]. A failure in this equilibrium may lead to one of two scenarios: (Scenario 1) an excess in vacant vehicles and competition; (Scenario 2) larger waiting times for passengers and lower taxi reliability. Consequently,the following question arises: Is it possible to guarantee that the taxi's spatial distribution over time will always meet the demand?

The taxi driver mobility intelligence is an important factor to maximize both profit and reliability within every possible scenario. Knowledge on where the services (transporting a passenger from a pick-up to a drop-off location) will actually emerge can be an advantage for the driver - especially when there is no economic viability of adopting random cruising strategies to find passengers. The GPS historical data is one of the main variables of this topic because it can reveal underlying running mobility patterns.

Hereby, we are focused on the real-time choice problem of which is the best taxi stand to go to after a passenger drop-off (i.e. the stand where another passenger can be picked-up more quickly). An intelligent approach regarding this problem will improve network reliability for both companies and clients: an intelligent distribution of vehicles throughout stands will reduce the average waiting time to pick-up a passenger while the distance traveled will be more profitable (by increasing the ratio between vacant and occupied cruising time). Furthermore, whenever they need a taxi, passengers will also experience a lower waiting time to get a vacant taxi (automatically dispatched or directly picked-up at a stand). This is a true advantage for a fleet competitively to its competitors.

The stand-choice problem is based on four key variables: (i) the expected revenue for a service over time, (ii) the distance/cost relation with each stand, (iii) the number of taxis already waiting at each stand and (iv) the passenger demand for each stand over time. However, at the best of our knowledge, there is no work handling this recommendation problem by using these four variables simultaneously (see Section 2.2.2 to know more about this topic). In this thesis, we argue that the taxi vehicular network can be a ubiquitous sensor of taxi-passenger demand from where the abovementioned variables can be continuously mined. The variable (iii) can be directly computed by the real-time vehicle's position - however, the remaining three need to be estimated for a short-term time horizon. The variables (i,iv) are handled in this Chapter while the variable (ii) is addressed in the next one.

This Chapter presents a model to estimate **the short-term demand that will emerge at a given taxi stand**. Specifically, it depicts the demand over space (taxi stand) for a short-time horizon of P-minutes. Such demand can be decomposed into two axis: the (iv) pick-up quantity (i.e. an integer representing the number of services to be demanded) and (i) the expected revenue for a service over time (i.e. a fare-based category). To do it so, this framework relies on both time series analysis and discretization techniques which are able to perform such supervised learning task **incrementally**.

The remainder of this Chapter is structured as follows. The Section 6.1 firstly describes how the dataset used was acquired and preprocessed. Then, some statistics about it are presented. The second Section highlights the literature gaps that are filled by the work described in this Chapter. Section 6.3 formally presents the methodology employed to carry out this predictive task. Section 6.4 describes how the methodology was tested in a real scenario: firstly, the experimental setup and metrics used to evaluate the model are introduced; then, the results obtained are presented in detail, followed by some important remarks. Finally, conclusions are drawn.

The symbols and notations used throughout this Chapter are provided in Table 6.1.

6.1 Data Preparation

The stream events data of a taxi company operating in the city of Porto, Portugal, was used as case study. This city is the center of a medium-sized urban area (consisting of 1.3 million inhabitants) where the passenger demand is lower than the number of running vacant taxis, resulting in a huge competition between both companies and drivers. According to a recent aerial survey of the road traffic of the city [Ferreira *et al.*, 2009], taxis represent 4% of the running vehicles during a non-rush hour period. The existing regulations force the drivers not to run *randomly* in the search for passengers; instead, they must choose a specific taxi stand out of the 63 existing ones in the city and to wait for the next service immediately after the last passenger drop-off. A map of the stands' spatial distribution is presented in Fig. 6.1.

There are three main ways to pick-up a passenger: (1) a passenger goes to a taxi stand and picks-up a taxi – the regulations also force the passengers to pick-up the first taxi in line (First In, First Out); (2) a passenger calls the taxi network central and requests a taxi for a specific location/time – the parked taxis have priority over the running vacant ones in the central taxi dispatch system; (3) a passenger picks a vacant taxi while it is going to a taxi stand, on any street.

Next Section describes the company studied, the data acquisition process, the preprocessing method applied as well as some descriptive statistics on such data.

6.1.1 Data Acquisition and Preprocessing

The data was continuously acquired using the telematics installed in each one of the 441 running vehicles of the company fleet. This taxi central usually runs in one out of three 8h shifts: midnight to 8am, from 8am to 4pm and from 4pm to midnight. Similarly to the dataset employed on Chapter 5 and described in Section 5.1, we also take advantage of the continuous GPS trace of each vehicle broadcasted with a given time periodicity. Each data chunk arrives each 5 seconds containing the following four attributes: (1) Vehicle Status (i.e. vacant, heading to pick-up a passenger, busy, etc.), a timestamp and the two GPS coordinates.

Based on such raw data, a novel trip-based dataset is built. Each sample corresponds to one non-vacant taxi service. It contains the following nine attributes: (1) TYPE – relative to the type of event reported. It has four possible values: *busy* - the driver picked-up a passenger; *assign* – the dispatch central assigned a previously required service; *free* – the driver dropped-off a passenger and *park* - the driver parked at a taxi stand. Attribute (2) STOP is an integer with the ID of the related taxi stand. Attribute (3) TIMESTAMP is the date/time in seconds of the event, and attribute (4) TAXI is the driver code; attributes (5) and (6) refer to the LATITUDE and the LONGITUDE corresponding to the acquired GPS position while attributes (7) and (8) refer to the cruising distance (in meters) and the cruising time (in seconds), respectively. The ninth attribute is the service fare. Conversely to the remaining ones, these values are not an exact representation of the ground truth since the farebox

Figure 6.1: Taxi Stand spatial distribution.

Table 6.1: Notation and symbols employed along this Chapter.

s_i	i_{th} taxi stand/urban area of a given case study
C_i	number of taxi vehicles already parked in s_i
υ_i	waiting time to pick-up the next passenger in s_i
χ	constant cost of letting a vehicle waiting in line at a stand per unit of time
X_k	discrete time series with the aggregated pick-up quantities on the taxi stand k
P	aggregation period of X_k
N	total number of existing taxi stands
$\lambda(t)$	time-varying expected value of pick-up quantities in place on a Poisson distribution
$d(t)$	weekday 1=Sunday, 2=Monday, ... of $\lambda(t)$
$\delta_{d(t)}$	relative change imposed by the weekday $d(t)$ on $\lambda(t)$
$h(t)$	period when time t falls (e.g. the time 00:31 is contained in period 2 for 30-minutes periods)
$\eta_{d(t),h(t)}$	relative change for the period $h(t)$ in the day $d(t)$ (e.g. the peak hours)
ω	weight set used on the terms of the Weighted Time Varying Poisson Model
γ	size of sliding window employed to compute ω
α	$0 < \alpha < 1$ smoothing factor of the Exponential Smoothing model employed to compute ω
Y_k	discrete time series with the aggregated pick-up quantities on the taxi stand k with a second level of aggregation used to drift X_k for on-demand pick-up quantity predictions
τ	aggregation period of Y_k
$R_{k,t}$	numerical prediction about pick-up quantity in the taxi stand k on the time instant t
ϕ, κ	optimal weight sets employed on the distinct terms of the ARIMA model
p, q	sizes of the weight sets ϕ, κ
θ	size sliding window employed to compute the ARIMA model
Δw	residual-based update performed over ϕ, κ to approximate their optimal value incrementally
β	user-defined parameter to set the reactivability of the incremental update rule Δw
$r_{k,t}$	ordered vector containing the revenue values corresponding to the amount paid by each service which started at the stand k during the time period $[t-1, t]$
$h(F, B)$	equal-width histogram employed to approximate the short-term fare p.d.f.
φ	number of bins of the $h(F, B)$
F	frequency set of $h(F, B)$
B	break points of $h(F, B)$
b_i	i_{th} break point of $h(F, B)$
μ	interval width of $h(F, B)$
mi, ma	minimum and maximum value of $h(F, B)$
l	number of predictive models employed in the ensemble
E_t	Error-based Ensemble of the predictive models about the short-term pick-up quantity prediction
H	sliding window size employed on the calculus of E_t
Q_t	Ensemble of the predictive models about the short-term fare-based p.d.f. estimation
ϱ	sliding window size employed on the calculus of Q_t
W	minimum radius to consider a service demand on a given taxi stand
ζ	generic error metric
$AG_{\zeta,t}$	aggregated error metric given by a weighted average of the error measured in all stands within the period $\{1, t\}$
ψ_k	total of services requested at the taxi stand k

transactions are not part of the studied data stream. To tackle this issue, an estimation model was developed based on a simplified version of Porto's taxi service price structure. It is illustrated in Table 6.2.

Table 6.2: Porto's taxi service price structure. Both the temporal and spatial fractions cost 0.15 euros.

Location	**Time**	**Minimum Price**	**Minimum Distance**	**Spatial Fraction**	**Temporal Fraction**
Inside the city limits	6am $\rightarrow$ 9pm	2.00eur.	220.0m	333.3m	37.0 sec.
	9pm $\rightarrow$ 6am	2.50eur.	176.0m	277.7m	37.0 sec.
Outside the city limits	6am $\rightarrow$ 9pm	3.25eur.	220.0m	166.6m	37.0 sec.
	9pm $\rightarrow$ 6am	3.90eur.	176.0m	138.9m	37.0 sec.

A time series of taxi demand services aggregated for a period of P-minutes was developed to handle the pick-up quantity prediction problem. There are three types of events: (1) the *busy* set directly at a taxi stand; (2) the *assign* set directly to a taxi parked at a taxi stand and (3) the *busy* set while a vacant taxi is cruising. Both a type 1 and type 2 events were considered as service required. However, for each type 2 event, the system receives a *busy* event a few minutes later – as soon as the driver effectively picks-up the passenger – which is ignored by our system. Type 3 events are ignored unless they occur in a radius of W meters from a taxi stand (where W is a user defined parameter). If it does, it is considered a type 1 event related to the nearest taxi stand according to the defined criteria. This was done because many regulations prohibit passengers from being picked-up in a predefined radius around a stop (in Porto, a 50m radius is in place).

6.1.2 Data Analysis

Statistics about the period studied are presented. Fig. 6.2 presents the sample distribution of the cruise time of the services demanded. Table 6.3 details the number of taxi services demanded per daily shift and day type in the two case studies. Additionally, we could state that, in both cases, the central service assignment is 24% of the total service (*versus* the 76% of the one demanded directly in the street) while 77% of the service demanded directly in the street is demanded in the stand (and 23% is assigned while they are cruising).

The average waiting time (to pick-up passengers) of a taxi parked at a taxi stand is 42 minutes while the average time for a service is only 11 minutes and 12 seconds (check the figure 6.2 to know more about the frequency distribution of the cruise time with passengers).

Fig. 6.3 represents three sample-based normalized estimations of the revenue's p.d.f.: a global estimation, one for the daytime revenues and another for the night time revenues. All estimations exhibit a **bimodal** structure. That is even clearer when the nighttime scenario is analysed. The time lag between the night time and the remaining p.d.f. indicates that the night time services usually have larger revenues than daytime services. Fig. 6.4 illustrates an equal-width revenue histogram and its cumulative frequency. Note that nearly 60% of the demanded services have a revenue below 6 euros. This pattern shows how difficult it is to maintain a balanced relationship between service offer and demand in this particular case study.

These statistics reflect the current economic crisis in Portugal and the inabil-

Table 6.3: Taxi Services Volume (Per Daytype/Shifts on a Daily Basis).

Daytype Group	**Total Services Emerged**	**Averaged Service Demand per Shift**		
		0am to 8am	8am to 4pm	4pm to 0am
Workdays	957265	935	2055	1422
Weekends	226504	947	2411	1909
All Daytypes	1380153	1029	2023	1503

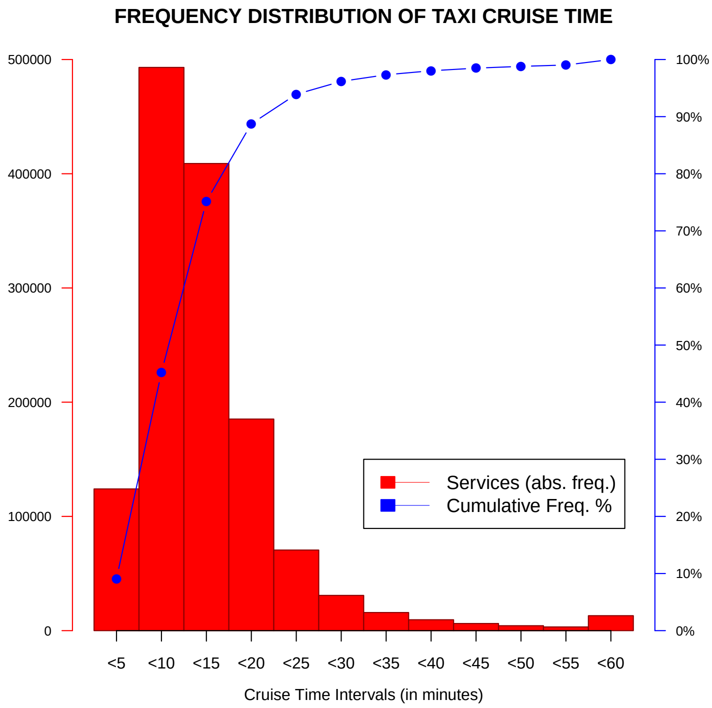


Figure 6.2: Frequency Distribution of Taxi Cruise Time for the entire data set.

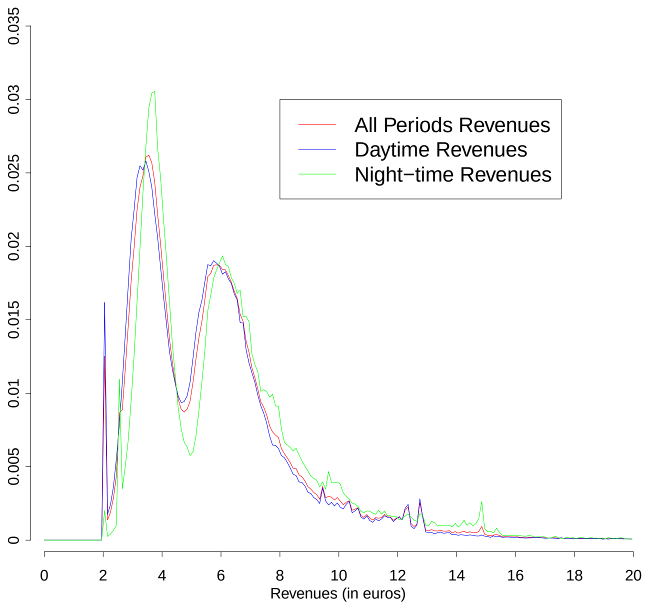


Figure 6.3: Normalized Kernel Density Estimation for the revenues p.d.f. detailed by daytime and night time.

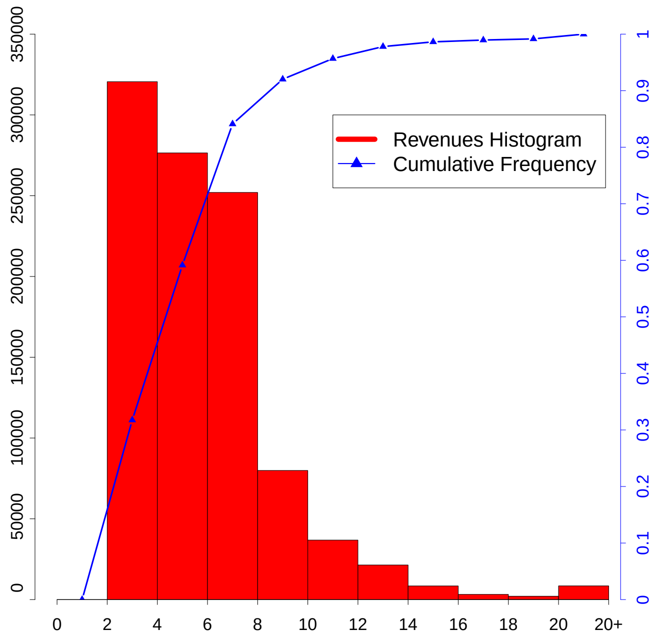


Figure 6.4: Revenue Histogram and its cumulative frequency. Left side y-axis refers to each bin's frequency (absolute) while the right one reflects the cumulative frequency (in blue).

ity of the regulators to reduce the number of taxis in Porto. It also highlights the importance of a recommendation system, where the shortness of services could be mitigated by getting services from the competitors.

6.2 Related Work

More and more datasets containing historical GPS data sets are being explored to improve taxi driver profitability. Typically, studies just handle this problem by handling the short-term demand. To do so, one of the two following approaches are taken: (1) predicting the number of service requests within a given area or (2) selecting some areas where there will be a high demand for services in the short-term. This is a relatively new topic. In (1), the State-of-the-Art approaches are time series analysis techniques [Li *et al.*, 2012] - however, there are many issues to handle on this topic. Such issues include to (1-i) let these models evolve continuously (instead of computing each prediction for fixed time spans) and also to (1-ii) use, somehow, the long term historical data into our learning model (check Section 2.2.3 to know more about this topic).

Type-2 approaches rely on recommending highly profitable routes. The main goal of these routing techniques is to establish Origin-Destination matrices to select demand hotspots (regions that are more likely to provide high demand rates). Such problem is typically performed using unsupervised learning techniques (namely, Spatial Clustering). Hierarchical clustering is employed in [Yue *et al.*, 2009; Chang *et al.*, 2010a] while Chang *et al.* [2010a] explores DBSCAN [Ester *et al.*, 1996] to mine time-dependent attractive areas by analyzing the historical time series of demands within predefined time spans. These approaches were extended by Hu *et al.* [2012] who proposed a heuristic function to connect the centroids of the top-k hotspots and a probability model to estimate the gasoline consumption in every route to compute the link weights.

The abovementioned approaches aim at increasing the ratio between live and cruising miles. However, this may be misleading as the variability in service revenue is high, especially from region to region [Powell *et al.*, 2011]. Let us formulate this issue with a numerical example: a predictive model of interest forecasted a demand of $d_1 = 10$ and $d_2 = 6$ services in areas/stands s_1, s_2, respectively, during the following period of P minutes. Let C_1, C_2 denote the number of cars already parked in the stands. The profit at each stand can be expressed as follows:

$$s_{profit} = r - \Big(\frac{C \times P}{d} \times \chi\Big) \tag{6.1}$$

where r is the expected service revenue and χ expresses the **constant** cost of letting a vehicle wait in line at a stand per unit of time. Assuming that both are equally distant from our current location and that the number of vehicles already parked in those areas is similar (i.e. $C_1 \sim C_2$), it is possible to estimate the relationship between waiting times to pick-up the next passenger at each stand υ_1, υ_2 as $\upsilon_1 = 0.6 \times \upsilon_2$. Considering the waiting time cost independent from the area under analysis and the average revenue at each stand r_1, r_2 being, for instance, \$8 and \$14, respectively, the most profitable stand would be s_2 and not s_1. A typical example of this could be airports, where long-runs are normally provided from city outskirts to downtown areas.

Li *et al.* [2012] presented a more accurate approach to the profitability problem by profiling the driver's experience according to the historical data on high-profit/low-profit drivers. Powell *et al.* [2011] presents a model to estimate the most profitable route by employing a spatial window to model the profitability of the neighboring regions, regarding the short-term decision on the path to take. The area's revenue scores end up being computed based on a moving average of the fares using a very short time window (i.e. 60 minutes). Notwithstanding their contributions, the authors oversimplify the concept of "low/high" fare by maintaining the fares as continuous variables - which forces to adopt a constant threshold between both categories. Such threshold can be misleading (e.g. a \$10 dollars service may not be relevant on the morning peak but can be valuable on the evening one; a peak value can be harder to predict than a class label).

By maintaining a fair approximation to the revenue p.d.f., the approach introduced in this thesis is **adaptable** to every scenario, allowing the user to decide which should be the rules in place to consider a service revenue *high*. Moreover, it combines sliding windows of different lengths to explore the historical data on different levels. The methodologies to do such demand estimation are presented in the following Section.

6.3 Methodology

Let $X_k = \{X_{k,0}, X_{k,1}, ..., X_{k,t}\}$ be a discrete time series (aggregation period of P-minutes) for the number of demanded services at a taxi stand k. The first goal is to build a model which determines the (1) set of service counts $X_{k,t+1}$ for instant $t+1$ and per taxi stand $k \in \{1, ..., N\}$. Then, we propose to build a short-term estimation of the fare-based p.d.f. for each taxi stand k using its series of service counts (i.e. pick-up quantities X_k). The methodology proposed to handle such problems is described throughout this Section.

6.3.1 Poisson Processes

Time Varying Poisson Model

This Section presents a model firstly proposed by Ihler *et al.* [2006]. The demand for taxi services exhibits, like other modes of road transportation (i.e. buses; check Section 4.1), a daily periodicity that reflects the patterns of the human activity. As result, the data appear to be non-homogeneous. Fig. 6.5 illustrates a one month taxi service analysis extracted from our dataset that illustrates this periodicity (the dataset is described in detail in Section 6.1).

Consider the probability for n taxi assignments to emerge in a certain time period - $p(n)$ - following a **Poisson Distribution**. It is possible to define it using the following equation

$$p(n; \lambda) = \frac{e^{-\lambda}\lambda^n}{n!} \tag{6.2}$$

where λ represents the rate (average demand for taxi services) in a fixed time interval. However, in this specific problem, the rate λ is not constant but time-variant. Therefore, it was adapted as a function of time, i.e. $\lambda(t)$, transforming the Poisson distribution into a non homogeneous one. Let λ_0 be the average (i.e. expected) rate of the Poisson process over a full week. Consider $\lambda(t)$ to be defined as follows

$$\lambda(t) = \lambda_0 \delta_{d(t)} \eta_{d(t),h(t)} \tag{6.3}$$

where $\delta_{d(t)}$ is the relative change for the weekday $d(t)$ (e.g.: Saturdays have lower day rates than Tuesdays); $\eta_{d(t),h(t)}$ is the relative change for the period $h(t)$ in the day $d(t)$ (e.g. the peak hours); $d(t)$ represents the weekday 1=Sunday, 2=Monday, ...; and $h(t)$ represents the period when time t falls (e.g. the time 00:31 is contained in period 2 if we consider 30-minutes periods).

Consider $\lambda(t)$ to be a discrete function (e.g.: an histogram time series of event' counts aggregated in periods of P minutes). The equation (6.3) requires the validity of both equations

$$\sum_{i=1}^{7} \delta_i = 7 \tag{6.4}$$

$$\sum_{i=1}^{I} \eta_{d,i} = I, \forall d \tag{6.5}$$

where I is the number of time intervals in a day. The result is discrete time series per stand representing the expected demand during an entire week: $\lambda(t)_k$. Each value in this series is an average of all demands previously measured in the same daytype and period (i.e. the expected service demand for a Monday from 8:00 to 8:30 is the average of the demand on all past Mondays from 8:00 to 8:30).

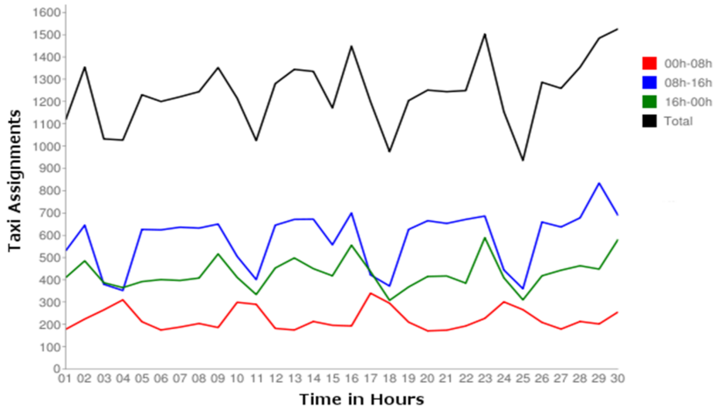


Figure 6.5: One month data analysis (total and per shift).

Weighted Time Varying Poisson Model

The model previously presented can be seen as a time-dependent average which produces predictions based on long-term historical data. However, it is not guaranteed that every taxi stand will have a highly regular passenger demand: in fact, the demand in many stands can often be **seasonal**. The beaches are a good example of the seasonality demand as taxi demand will be higher during summer weekends as opposed to other seasons throughout the year.

To face this specific issue, a weighted average model is adopted. Its definition is based on the model presented before: the goal is to increase the relevance of the demand pattern observed in the recent week (e.g. what happened on the previous Tuesday is more relevant than what happened two or three Tuesdays ago). The weight set ω is calculated using a well-known time series approach to these type of problems: the Exponential Smoothing [Holt, 2004]. It is possible to define ω as follows

$$\omega = \alpha * \{1, (1-\alpha), (1-\alpha)^2, ..., (1-\alpha)^{\gamma-1}\}, \gamma \in \mathbb{N} \tag{6.6}$$

where γ is the number of historical periods considered and $0 < \alpha < 1$ is the smoothing factor (i.e. γ and α are user-defined parameters). Then, based on the previous definition of $\lambda(t)_k$, it is possible to define the resulting weighted average $\mu(t)_k$ as follows

$$\mu(t)_k = \sum_{i=1}^{\gamma} \frac{X_{t-(\theta*i)} * \omega_i}{\Omega}, \Omega = \sum_{i=1}^{\gamma} \omega_i \tag{6.7}$$

where θ is the number of time periods contained in a week.

On Maintaining Histograms Incrementally

These two methods are clearly able to deal to unbounded streams of data: the first one is *incremental* because it is possible to maintain the averages additively by keeping in memory/database just the number of periods considered and the average of services measured so far; the second one works with a sliding window of γ weeks, discarding the remaining examples. However, the service counts

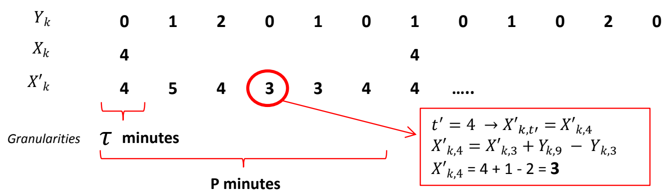


Figure 6.6: An example about how can we additively calculate one term of the series $X'_{k,t}$.

are somehow *stuck* to the bin boundaries (i.e. the start and end time points defined for each bin). Supposing that we maintain a daily histogram of $P = 30$ minutes and it starts at midnight but we want to produce predictions with a periodicity of $\tau = 10$ minutes (the problem can be generalized to all $\tau \neq P$. The histogram will start at midnight and each bin will have the counts for the service demanded for periods of 30 minutes. How can we have the information about what happened between 09:20am and 09:50am if the available bins have just counts from 09:00am to 09:30am and so on? Is it necessary to maintain *every possible* histograms in memory? In this Section, we present a solution to overcome this issue.

One of the main ways to handle this type of problems is to perform an **incremental discretization** (see, for instance, [Gama and Pinto, 2006]). An event count X_t in an interval $[t, t+P]$ will be very *similar* to the count X_{t+1} in the interval $[t+\tau, t+P+\tau]$ (as much as $\tau \sim 0$). We can formulate it as

$$X_{t+1} = X_t + X'_{[t+P,t+P+\tau]} - X'_{[t,t+\tau]} \tag{6.8}$$

where X' represents both the continuous event count on the first τ-minutes of the interval $[t, t+P]$ and on the τ-minutes immediately after the same period. Consequently, it is possible to define two discrete time series of services demand on a taxi stand k as $X_k = \{X_{k,0}, X_{k,1}, ..., X_{k,t}\}$ and $Y_k = \{Y_{k,0}, Y_{k,1}, ..., Y_{k,t'}\}$ (where $t' < t$) using granularities of P and τ minutes, respectively. Let X'_k be the discrete time series needed to predict the event count on the interval $[t', t'+\tau]$. We can define the event count at the time period $[t', t'+P]$ as following

$$X'_{k,t'} = \begin{cases} X'_{k,t'-1} + Y_{k,t'+P/\tau-1} - Y_{k,t'-1} \text{ if } t' > t \\ X_{k,t} \text{ if } t' = t \end{cases} \tag{6.9}$$

We take advantage of the *additive* characteristics of both time series to rapidly calculate a new series of interest maintaining two aggregation levels/layers: P and τ. An illustrative example about how this series can be calculated is presented in Fig. 6.6.

6.3.2 AutoRegressive Integrated Moving Average Model

The AutoRegressive Integrated Moving Average Model (ARIMA) [Box *et al.*, 1976] is a well-known methodology to both model and forecast univariate time series data such as traffic flow data [Min and Wynter, 2011], electricity price [Contreras *et al.*, 2003] and other short-term prediction problems such as the one presented here. A detailed description about this method is already presented in Section 3.3.2. Let $R_{k,t}$ be a numerical prediction about pick-up quantity in the taxi stand k on the time instant t. Given the historical time series of events, X_k, we can formulate the underlying process that generates the time series (taxi service over time for a given stand k) based on the general ARIMA equation (3.10), as

$$\begin{aligned} R_{k,t} = \kappa_0 + \phi_1 X_{k,t-1} + \phi_2 X_{k,t-2} + ... + \phi_p X_{k,t-p} \\ +\varepsilon_{k,t} - \kappa_1 \varepsilon_{k,t-1} - \kappa_2 \varepsilon_{k,t-2} - ... - \kappa_q \varepsilon_{k,t-q} \end{aligned} \tag{6.10}$$

A study conducted on time series from the demand of taxi services in one of the busiest taxi stands is presented in Fig. 6.7.

Despite its utility, the ARIMA method is basically an offline method because it requires the availability of all data points to compute its prediction. This issue is commonly overcame by introducing a sliding window outside which the data points are simply discarded (in this work, we also employ such approach; the sliding window size used to do so is described in Section 6.4.2). Even so, its optimality is highly correlated with the weights sets $\kappa_m (m = 0, 1, 2, ..., q)$ and $\phi_l (l = 1, 2, ..., p)$ - the weights set to each data point of both autoregressive and moving average components (check the equation (6.10)). In other words, it is usually necessary to *fit* such weight set to the past data points each time we want to do a prediction [Min and Wynter, 2011; Contreras *et al.*, 2003]. The computation of such optimal weight set may be heavy - specially if we consider a high periodicity in our system (i.e $\tau << P$). A way to avoid such computational effort is presented below.

An *Incremental* ARIMA Model

The ARIMA model relies on calculating the present event count using a linear combination of previous samples. In eq. 6.10 the $\phi_l (l = 1, 2, ..., p)$ and $\kappa_m (m = 0, 1, 2, ..., q)$ are the model weights. Such weights usually need to be fitted using the entire historical time series every time we build a new prediction. This operation can represent a high computational cost if we employ it at a such large scale as we do here. Similarly to the Bus Bunching problem presented in

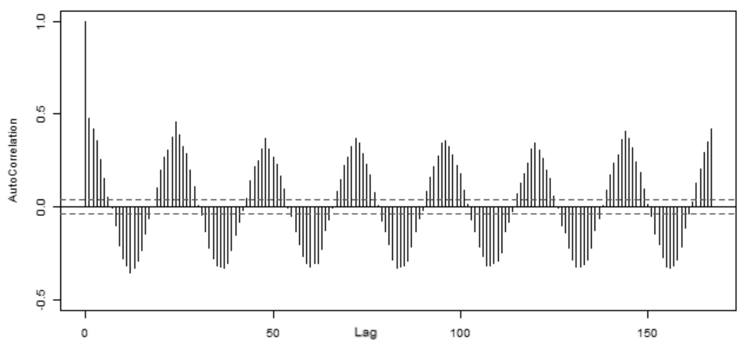


Figure 6.7: Autocorrelation profile for data on the demand for taxi service obtained from one of the busiest taxi stands in the city. The x-axis has different period lags studied and the y-axis contains the correlation within the signal. Note that there are peaks for each 12h periods.

the previous Chapter, there is not much time to do such optimal training of the weight set (especially for low values of τ).

To overcome this issue, we propose to use the **Linear delta rule** (described in Section 5.3.2) to update these weights recursively instead of re-calculating them iteratively as we did so far. This rule consists of updating the weights by increasing/decreasing them using a *direct proportion* of the difference between the predicted and the real output. Consider $R_k = \{R_{k,1}, R_{k,2}, ..., R_{k,t}\}$ to be a time series with the number of services predicted for a taxi stand of interest k in the period $[1, t]$ and X_k be the number of services actually emerged in the same conditions. Let $w_{k,t} = \{w_{k,t,1}, w_{k,t,2}, ..., w_{k,t,z}\}$ be a set of z weights of a predictive model of interest (like ϕ and κ in the ARIMA one) used to calculate $R_{k,t}$. Departing from eq. 5.8, it is possible to calculate the update set $\Delta w_{k,t} = \{\Delta w_{k,t,1}, ..., \Delta w_{k,t,j}\}$ as follows

$$\Delta w_{k,t,j} = \beta(R_{k,t} - X_{k,t})w_{k,t,j}, \forall j \in \{1, ..., z\} \tag{6.11}$$

where β is an user-defined proportionally constant which sets how reactive the model should be. This way, the ARIMA weights can be *incrementally* updated.

6.3.3 Fare-based p.d.f. estimation

Let $r_{k,t}$ denote a vector containing the revenue values corresponding to the amount paid by each service which starts at the stand k at time period t, where $X_{k,t} = |r_{k,t}|$. To characterize the distribution of these values, we propose to approximate its local p.d.f.. One of the best known ways of doing that is by discretizing the variable values into intervals using histograms [Gama and Pinto, 2006]. By dividing the number of services $X_{k,t}$ into φ bins according to service revenue, it is possible to obtain φ discrete time series for the number of services requested within a certain *revenue interval*. Secondly, a set of fixed rules is employed to classify the period's profitability based on those histograms. Thirdly, the time series analysis above described are employed to estimate the future frequencies of these φ bins. Those values are used to predict the stand's short-term profitability class by employing the abovementioned set of rules.

The first goal is to discretize the revenues into a value *interval* $\pi_i = [b_i, b_{i+1}) \in \Pi$ for $r_{k,t}$ such that $b_i \leq r_{k,t} < b_{i+1}$. Π can be defined as follows

$$\Pi = \{\pi_i | \pi_i = [b_i, b_{i+1}) : b_{i+1} - b_i = b_i - b_{i-1}, \forall b_i \in N\} \tag{6.12}$$

where $\xi = b_{i+1} - b_i$ denotes the interval **width**. Consequently, it is possible to obtain an *equal-width* histogram $h(F, B)$ defined by the aforementioned set of break points $B = b_1, ..., b_{\varphi-1}$ and a set of frequency counts $F = f_1, ..., f_{\varphi}$. To define the number of bins φ, it is necessary to define the *range* of the random variable and the desired *interval width*. For that, three user-defined parameters are employed: the interval width μ and a minimum/maximum value as mi, ma, respectively. Therefore, it is possible to redefine π_i as follows:

$$\pi_i = [mi + \mu \times (i-1), mi + \mu \times i) : (mi + \mu \times i) \leq ma \tag{6.13}$$

An additional *last* bucket is added to the ones defined in Π to account for all the revenue values above the threshold value (i.e. ma). Consequently, $\varphi = |\Pi|+1$.

By employing these histograms, it is possible to monitor the evolution of the revenue's p.d.f. at a given taxi stand to predict the short-term one. Estimating the p.d.f. estimation brings a vast range of possibilities when it comes to building a set of rules (or multiple rules) capable of classifying the stand's profitability in every time period. The set of rules used in this particular scenario is described in Section 6.4.1.

Regardless of the evolution of the p.d.f. throughout time, the number of bins φ is constant over time (it only depends on the parameters ma, mi and μ). Consequently, each bin can be seen as a time series in terms of the number of services requested at that stand where the revenues are constrained by a given interval. This observation makes it possible to model the p.d.f. estimation problem as multiple time series forecasting ones. The three predictive frameworks described along the Sections 6.3.1 and 6.3.2 can be used to perform three distinct predictions on this variable.

6.3.4 Sliding Window Ensemble Framework

Three distinct predictive models have been proposed which focus on learning from the long, medium and short-term historical data. However, a question remains open: Is it possible to combine them all to improve our prediction? Over the last decade, regression and classification tasks on streams attracted the community attention due to the need to adapt these supervised learning models to the concept drifts that are constantly introduced in the data. The ensembles of such models are one of the ways to handle with such drifts. One of the most popular ensemble models is the weighted ensemble [Wang *et al.*, 2003]. Two ensemble models are proposed to handle the service count prediction (i.e. E_t) and the fare-based p.d.f. estimation (i.e. Q_t), respectively[1]. Both are based on such weighted ensemble framework. They are described along this Section.

Consider $M = \{M_1, M_2, ..., M_l\}$ to be a set of l models (i.e. hereby, $l = 3$) of interest to model a given time series and $G = \{G_{1t}, G_{2t}, ..., G_{lt}\}$ to be the set of forecasted values for the pick-up quantities during the next period on the interval t by those models. The ensemble forecast E_t is obtained as

$$E_t = \sum_{i=1}^{l} \frac{G_{it} * (1 - \rho_{iH})}{\Upsilon}, \Upsilon = \sum_{i=1}^{l} (1 - \rho_{iH}) \tag{6.14}$$

where ρ_{iH} is the error of the model M_i in the periods contained on the time window $[t-H, t]$ (H is a user-defined parameter to define the window size) comparatively to the real service count time series. As the information is arriving continuously for the next periods $t, t+1, t+2, ...$ the window will also **slide** to determine how the models are performing in the **last H periods**. To calculate such error, the Symmetric Mean Percentage Error (*sMAPE*) was used (as it is further discussed in Section 6.4.2).

Let $Gf = \{Gf_{1t}, Gf_{2t}, ..., Gf_{lt}\}$ be the set of forecasted fare-based labels during the next period on the interval t by the models in M. A majority

[1] Note that both series also depend on the taxi stand k. However, this notation was omitted in this subsection for simplify its comprehension.

voting scheme was employed to combine the label outputs according to each prediction. This simple scheme consists of measuring the **average accuracy** of each method on the **last ϱ periods** - where ϱ is an user-defined parameter.

6.4 Experiments

This Section starts by describing the experimental setup developed to test the model on the available data. Secondly, the metrics used to evaluate the methods are enumerated. Finally, the results achieved are presented.

6.4.1 Experimental Setup

Similarly to the work described in the previous Chapter, the test-bed employed in this study also followed a prequential evaluation scheme. Consequently, two sliding windows were used to measure the models' error before each new demand prediction (i.e. H for the pick-up quantities and ϱ for the fare-based prediction). The metrics used to do so are defined in Section 6.4.2.

Each data chunk was transmitted and received through a socket. The predictive models were implemented using the R language [R Core Team, 2012]. The prediction effort was divided into three distinct processes running on a multi-core CPU (i.e. the time series for each stand is independent from the remaining ones), which reduced the computational time required for each forecast. Fig. 6.9 illustrates the test-bed described: the $PP_i...PP_t(t = 3)$ are the independent predicting processes – each one handles a predetermined group of taxi stands. The pre-defined functions used and the values set for the model parameters are described in detail along this Section.

An aggregation period of 30 minutes (i.e. forecast horizon of $P = 30$ minutes) and a radius of $W = 100$m ($W > 50$ is defined by the existing regulations) were set. This aggregation was set based on the average waiting time at a taxi stand, i.e. a forecast horizon lower than 42 minutes. These series were used with a fixed time-span on the fare-based demand prediction model. However, a new time series was built for the pick-up quantity prediction model. This was done to handle the demand peaks and valleys that often arises on some city areas. This time series has an aggregation period of 5-minutes ($\tau = 5$). It was created according to the definition presented in Section 6.3.1.

Both the ARIMA model (p, d, q values and seasonality) and the ARIMA weight sets ϕ and κ were firstly set (and updated each 24h) by learning/detecting the underlying model (i.e. autocorrelation and partial autocorrelation analysis) running on the historical time series curve of each stand during the last two weeks (i.e. period $t - 2\theta, t$). To do so, we used an automatic time series function from the `[forecast]` R package - *auto-arima* – and the *arima* function from the built-in R package `[stats]`. The weight set is then *incrementally* updated for each 24h period according with eq. (6.11).

A parameter tuning task was conducted on the parameters α and β based on a simplified version of Sequential Monte Carlo method (the reader can consult the survey in [Cappé *et al.*, 2007] to know more about this topic). The goal was to calibrate the model by finding the optimal subregion on the input space

$\alpha, \beta \in [0, 1]$ which maximizes the predictive performance. To do so, 100 distinct samples were generated as admissible values for α and they were tested using an older and smaller dataset containing data very similar to the one tested in our experiments (i.e. the same feature space). Since $\tau << P$, it is reasonable to admit that the ARIMA weight sets ϕ and κ will have short differences from prediction to prediction (i.e. $\beta \leq 0.1$). Therefore, 10 admissible values for β were considered with a step of 0.01 between each one of them satisfying the following inequation: $0 \leq \beta \leq 0.1$. All the possible combinations of these values of β and α were considered on these tuning tests.

As result, it was possible to determine the ideal values as $\alpha = 0.4$ and $\beta = 0.01$. These values demonstrated to be robust since small changes did not cause a relevant impact on the model output. These values remained stable on the following input space: 0.4 ± 0.1 and 0.01 ± 0.005 for α and β, respectively. Therefore, $\alpha = 0.4$ and $\beta = 0.01$ were used in the experiments. The γ value was set respecting the following definition

$$\gamma = \underset{\gamma}{arg\,min}\ \omega_\gamma : \omega_\gamma \geq 0.01, \gamma \in \mathbb{N} \tag{6.15}$$

Using this equation on our experimental setup, we obtained that $\alpha = 0.4 \implies \gamma = 8$. Using this very same experimental setup, we repeated the experiments using all the components of our method (i.e. the ensemble of our three predictive models) and we tested five possible values for the size of the ensemble sliding window, $H = \{2, 4, 6, 8, 10\}$. The best results were obtained for $H = 4$ and therefore, this was the value considered (i.e. it represents a sliding window of 20 minutes).

The parameter setting for (ma, mi, μ) resulted in histograms with three bins (i.e. $\varphi = 3$). For this particular scenario, we established a three-class set to estimate the stands' profitability (”*low*”,”*medium*” and ”*high*”). A user-defined set of rules was developed for this particular task adapted to the present scenario. Its pseudo code is displayed in Fig. 6.8. However, we do want to sustain that this approach can be adapted by any taxi network by changing φ, the number of classes and the rule set in place.

Table 6.4 summarizes the information about the learning periods used by each algorithm while Table 6.5 presents the values employed on the remaining parameters.

Table 6.4: Description of the Learning Periods.

Algorithm	**Sliding Window**	**Nr. of Periods Considered**
Poisson Mean	All Data $\{1, t\}$	N/A: it is calculated incrementally
Weighted Poisson Mean	Last two months	$\gamma = 8$
ARIMA	Last two weeks	$2 * \theta$
Pick-Ups Quantity Ensemble	Last twenty minutes	$H = 4$
Fare-based Label Ensemble	One day	$\varrho = 48$

```
function CLASSIFY-PERIOD(h(F, B), X_{k+t})
    if X_{k+t} = 0 then return "low";
    end if
    if X_{k+t} <= 5 then
        if b_1 = 0 then return "medium";
        else
            return "low";
        end if
    end if
    b1_{ratio} = b_1/X_{k+t};
    if b1_{ratio} < (1 - 0.4) then return "high";
    else
        if b1_{ratio} < (1 - 0.2) then return "medium";
        end if
    end if
    return "low";
end function
```

Figure 6.8: Algorithm for the Period's Profitability Classification using the Revenue Histogram. The parameters represent the histogram's frequencies (F) and break points (B), as well as its total mass X_{k+t}.

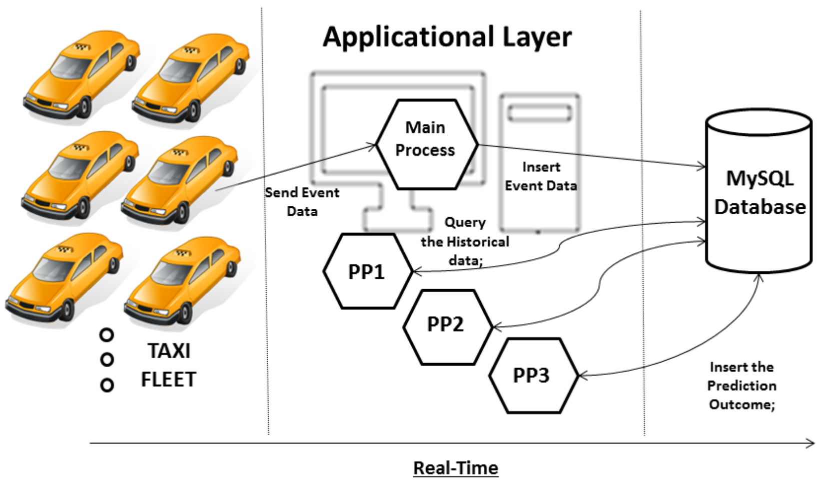


Figure 6.9: Illustration of the streaming test-bed.

6.4.2 Evaluation Metrics

We used the data obtained from the last four months to evaluate our framework on the pick-up quantity prediction problem. A well-known error measurement was employed to evaluate our output: the Symmetric Mean Percentage Error ($sMAPE$) (which was formally introduced in Section 3.5). Hereby, we independently evaluated the predictions for the pick-up quantities performed for each taxi stand (i.e. $sMAPE_k$). However, this metric can be too intolerant to small magnitude errors (e.g. if two services are predicted on a given period for a taxi stand of interest but no one emerges, the error measured during that period

Table 6.5: Parameter Setting used in the experiments.

Parameter	Value	Description
β	0.01	parameter to set the reactivability of the incremental update rule Δw
α	0.4	parameter to calculate the weight's curve on the Exponential Smoothing
P	30	aggregation period used to calculate the time series (in minutes)
τ	5	second-level aggregation period used to calculate the time series (in minutes)
mi	2	minimum value in the revenue histogram obtained for each period
ma	10	maximum bounded value in the revenue histogram obtained for each period
μ	4	bounded width of the intervals
W	100	minimum radius to consider a service demand on a given taxi stand (in meters)

would be 1). To produce more accurate statistics about series containing very small numbers, a Laplace estimator [Jaynes, 2003] is commonly added to eq. 3.21. In this case, we perform such normalization by adding a constant c to the denominator (i.e.: originally, it was added to the numerator to estimate a success rate [Jaynes, 2003]). The ($sMAPE_k$) (i.e.: the error measured on the time series of services predicted to the stand k) can be defined as

$$sMAPE_k = \frac{1}{t}\sum_{i=1}^{t}\frac{|R_{k,i} - X_{k,i}|}{R_{k,i} + X_{k,i} + c} \tag{6.16}$$

where c is a user-defined constant. To simplify the theorem application, we consider its most common use: $c = 1$ [Jaynes, 2003].

This metric is focused just on one time series for a given taxi stand k. However, the results presented below use an averaged error measure based on all stands series – AG. Consider ζ to be an error metric of interest. $AG_{\zeta,t}$ is an aggregated metric given by a weighted average of the error measured in all stands in the period $1, t$. It is formally presented in the following equations:

$$AG_{\zeta,t} = \sum_{k=1}^{N}\frac{\zeta_{t,k} * \psi_k}{\Psi} \tag{6.17}$$

$$\psi_k = \sum_{i=1}^{t} X_{k,i}, \Psi = \sum_{k=1}^{N}\psi_k \tag{6.18}$$

where ψ_k is the total of services requested at the taxi stand k; $\zeta_{t,k}$ is the error measured by ζ at the stand k and Ψ is the total of services emerged at all stands thus far.

The Accuracy (ACC) was used as evaluation metric on the fare-based stand classification problem. Moreover, the accuracy error was divided into `higher-prediction` and `lower-prediction` to discover when the predicted profitability is higher/lower than the real one. It was calculated for the N periods considered in the test set. They were then aggregated by calculating a *weighted* mean of their values at the existing taxi stands. Each stand's weight corresponds to the number of services requested on them.

6.4.3 Results

This Section starts by introducing the results obtained on the Pick-up Quantity Prediction problem. Then, the results obtained on the fare-based stand classification task are presented.

Table 6.6: Error Measured on the Stand-based Pick-up Quantity Prediction problem using $sMAPE$.

Model	Periods			
	00h–08h	**08h–16h**	**16h–00h**	**24h**
Poisson Mean	27.67%	24.29%	25.27%	25.32%
W. Poisson Mean	27.27%	24.62%	25.66%	25.28%
ARIMA	28.47%	24.80%	25.60%	26.21%
Ensemble	**24.86%**	**23.14%**	**24.07%**	**23.77%**

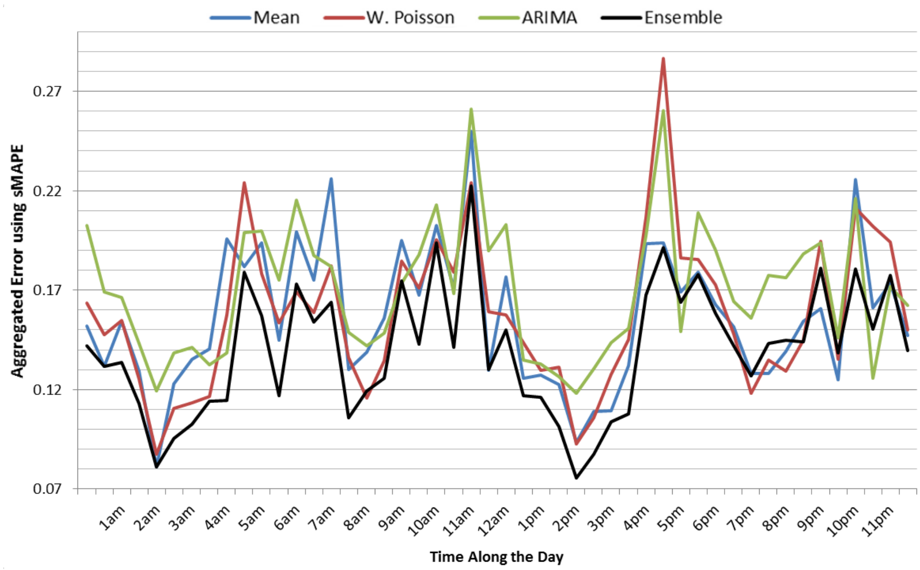


Figure 6.10: Pick-Up Quantity Ensemble evaluation on a typical Saturday. The results were aggregated from all stands.

Pick-up Quantity Prediction

The results on the pick-up quantity prediction are presented over four distinct perspectives: 1) averaged error of the proposed methods; 2) a comparative analysis of the ensemble performance versus the remaining models; 3) a direct analysis of output examples, and 4) a report on the computational time required to forecast the next period.

Firstly, the error measured for each model is presented in Table 6.6. The results are firstly presented per shift and then globally. The results were aggregated using the AG_β previously defined over the $sMAPE$ metric.

Secondly, Fig. 6.10 presents a comparison between our Ensemble and the other predictive models on a typical weekend day. These values were calculated using the same 20-minutes sliding window of the ensemble (the error of the instant t is the error measured at period $[t - H, t], H = 4$) with a periodicity of $P = 30$ minutes.

Thirdly, two distinct weekly analyses of the discrepancies between the demand predicted and the services actually provided are displayed in Fig. 6.11. It exhibits the demand prediction and its real outcome during a week in two distinct stands. It considers just the predictions made each period of $P = 30$ minutes.

The model forecasted the spatiotemporal taxi-passenger demand for a time horizon of 30-minutes with a periodicity of 5 minutes. It used (on average) 37.92 seconds (i.e. 0.607 seconds per time series/stand) using just one iterative process – one program, one CPU core:PP_1. The ARIMA model update was also fast: 48.12 seconds (mean value) were used to do so each 24h. Further offline experiments determined that such time can be reduced by 70% if we consider a parallel computing architecture like it is suggested in Fig. 6.9. These results are discussed next.

Fare-based Stand Classification

Fig. 6.12 presents descriptive statistics on the bin values of one of the busiest taxi stands in this case study. This statistics are divided by profitability class and also by day period. This division shows how the classification rule set (Fig. 6.8) works over the histograms. Using these rules, the following class distribution was obtained: *"low"*: 81.57%; *"medium"*: 13.10%; *"high"*: 5.33%. Table 6.7 presents a detailed evaluation of the five classification frameworks employed in this task. Finally, Fig. 6.13 divides the ensemble accuracy between each of the 63 taxi stands in Porto grouped with the number of services requested at the stand during the test period.

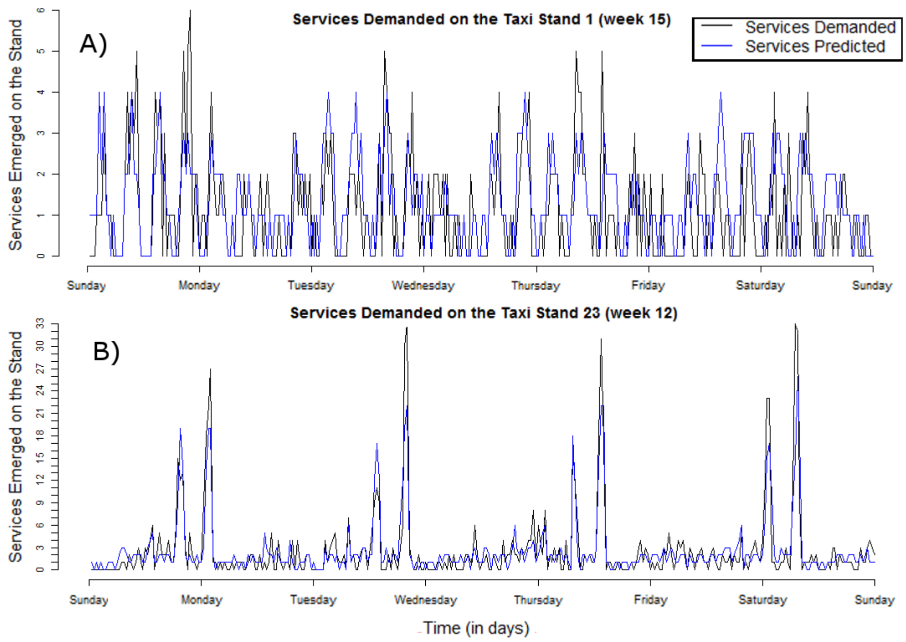


Figure 6.11: Weekly comparison between the services forecasted and the services emerged on two distinct scenarios regarding different taxi stands and weeks.

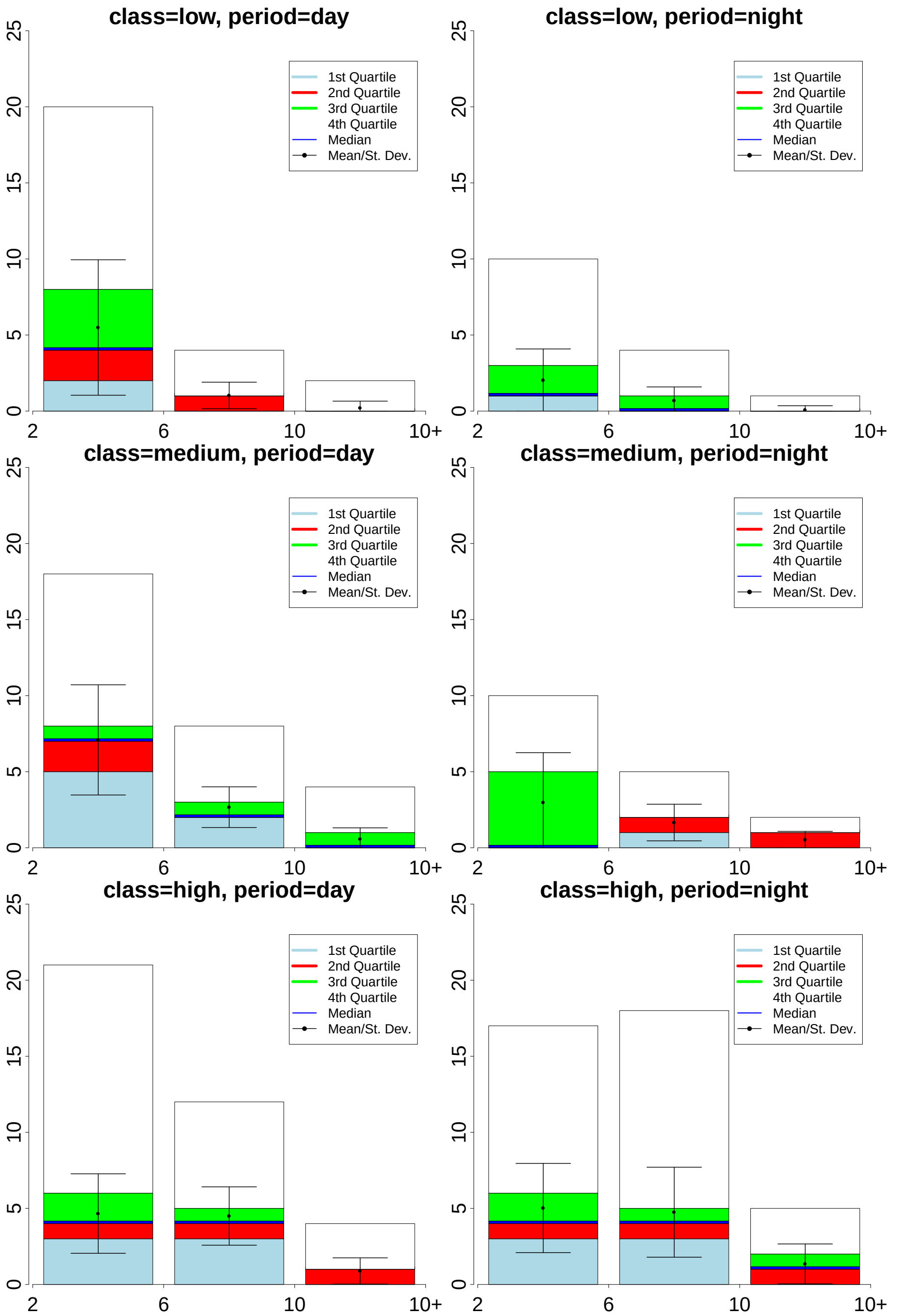


Figure 6.12: Descriptive Statistics on each bin values for different periods and profitability classes at taxi stand 57.

6.5 Discussion

The overall performance is very good: (1) the maximum value of the pick-up quantity error was 27.67% while (2) the profitability-based stand classification method surpasses the majority class - what is especially significant if we consider that we are facing an unbalanced classification task (i.e. 81.57% of the true labels are *"low"*. The sliding window ensemble is always the best model in every shift and period considered for both prediction tasks.

6.5.1 Pick-up Quantity Prediction

The ensemble methodology is robust comparatively to the remaining models: in Fig. 6.10 it is possible to identify a point where the ensemble maintained its performance while two other methods suffered a significant decrease in performance, highlighting the inherited learning of the ensemble approach. Fig. 6.11 presents two distinct scenarios to compare the forecasted and the real demand: in A), the demand corresponds to an irregular taxi stand where services do not have a usual pattern to emerge (even if the demand is low); in B), the chart corresponds to a completely regular stand behavior. The two examples illustrate that the ensemble can correctly forecast the demand in distinct scenarios, periods and time horizons.

In the present case study, the target variable is the number of services to arise at a taxi stand network during a pre-defined period of time. This variable was chosen due to the **stand** relevance in this scenario (where 76% of the total number of services is directly required to vehicles parked on them). However, this is not the reality in many large cities around the world due to their (de)regulation [Schaller, 2007]. Most authors in the literature on this topic divide their scenarios/urban areas into spatial clusters - as exemplified in Fig. 6.14 - to predict and/or characterize the pick-up quantity distribution on a short-term time horizon [Deng and Ji, 2011; Liu *et al.*, 2009; Yue *et al.*, 2009; Ge *et al.*, 2010; Yuan *et al.*, 2011b; Chang *et al.*, 2010b; Li *et al.*, 2012]. This mathematical model does not depend on how the services historical data are spatially aggregated (i.e. by stand or by spatial cluster) but only on the aggregation period of P-minutes (which is user-defined). Therefore, it also represents a straightforward contribution to previous work.

Table 6.7: Profitability Prediction Evaluation.

Method	**Accuracy**	**lower prediction error**	**higher prediction error**
Poisson Mean	73.63%	16.83%	9.54%
Exponential Smoothing	71.57%	16.66%	11.77%
ARIMA	70.91%	17.90%	11.19%
Majority Class	65.19%	22.18%	12.64%
Ensemble	**73.99%**	**17.88%**	**8.13%**

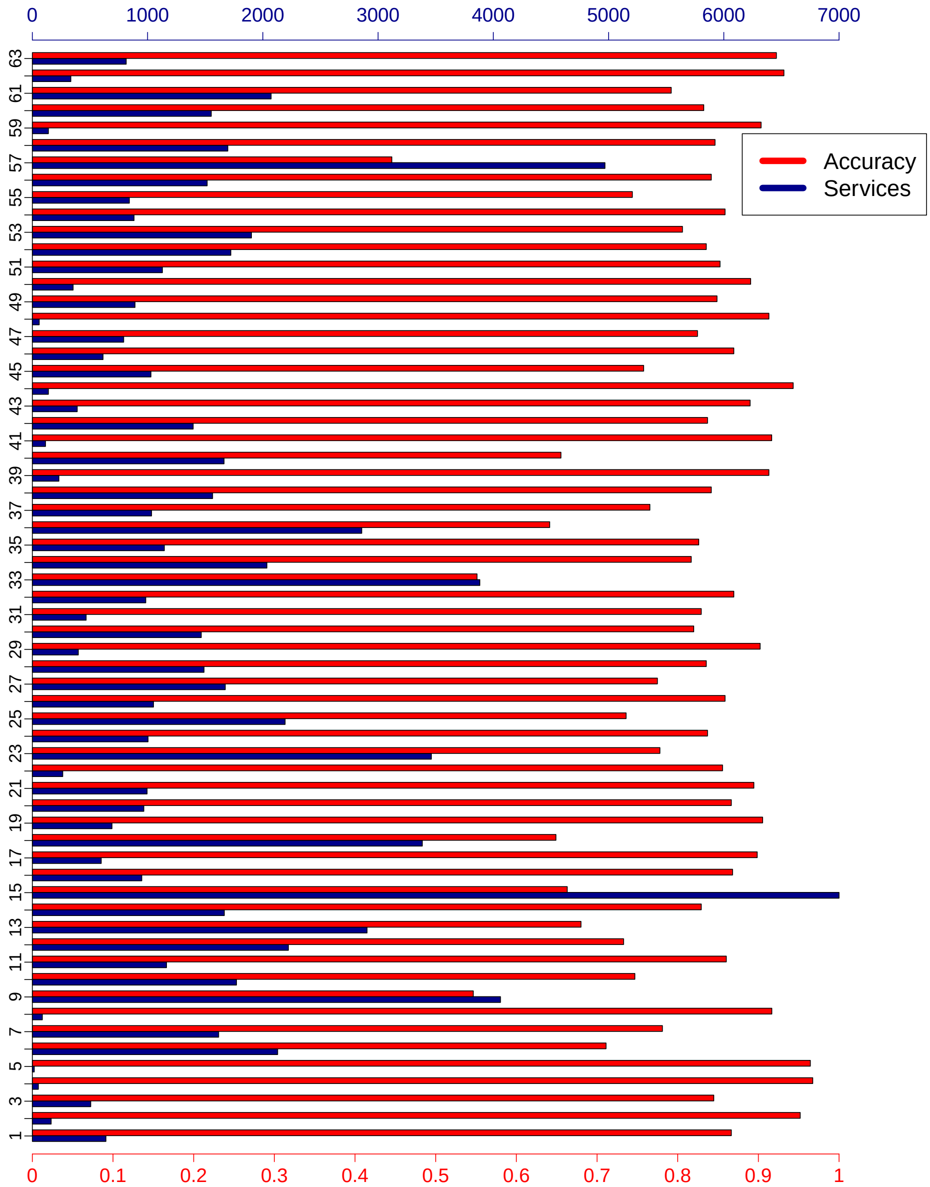


Figure 6.13: Ensemble evaluation detailed by stand. The grouped bars represent the accuracy (light red) and the number of services requested at each stand (dark blue).

6.5.2 Fare-based Stand Classification

Fig. 6.12 exemplifies the histograms distribution on distinct classes and scenarios. Note that the class (”low”/”medium”/”high”) does not have a direct relationship with the bins frequencies.

To approximate p.d.f. using histograms may seem quite simpler while compared with other estimation methods (e.g. kernel estimation). It may partially

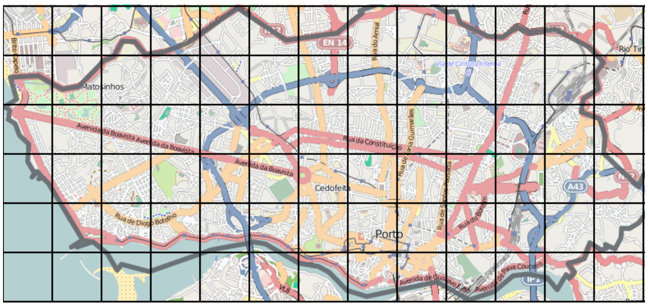


Figure 6.14: Example of a possible grid-based spatial clustering of the city of Porto, Portugal.

explain the accuracy errors on the stand revenue's classification. However, this method have a strong advantage facing the most common ones: it can be computed nearly **incrementally**, using one or just some of the most recent samples to estimate the next p.d.f..

The low number of bins (three) employed is a rough approximation of the true revenue p.d.f.. This level of detail is user-defined, along with the histogram classification rule set. The reduced length of the test set (i.e. one month) may not be enough to assume this setting as the best possible for this case study. Moreover, in more complex urban areas, it may be relevant to explore more complex p.d.f. approximations by determining which are the best parameter settings (i.e. ma, mi, μ and rule set) for each scenario. However, this discussion is not addressed in this thesis.

In Fig. 6.13, it is possible to observe that the ensemble method has an accuracy $\geq 90\%$ in most stands. The busiest stands present a lower accuracy than expected. This behavior may indicate that there is a persistent error on this type of stand. However, a stand-based analysis on the algorithm's behavior is required to reach these conclusions.

Despite the limitations mentioned above, this work is only a fair proof of concept for using the demand numerical predictions to uncover the stands' profitability. Note that nearly 70% of the classification error results in a profitability class that is **lower** than the period's true label. This shows how reliable this methodology can be by being **cautious** to predict high-revenues.

6.6 Conclusions

In this Chapter, we presented a **novel application of time series forecasting techniques to improve the taxi driver mobility intelligence**. It was done so by transforming both GPS and event signals emitted by taxi vehicles from a company operating in Porto, Portugal into time series of interest containing demand-based information. Secondly, time series analysis techniques were used to estimate the future values of these series. Then, these predictions were decomposed to build a fare-based p.d.f. able to classify each stand regarding the profitability of the services that will be demanded in a short-term. As a result, the model presented was able to predict the taxi-passenger demand at each one of the stands regarding both the pick-up quantities and the type of services to be demanded. It does it so with a short term horizon of $P = 30$ minutes and with a periodicity of $\tau = 5$ minutes.

The model presented a more than satisfactory performance, correctly predicting all the tested service with an aggregated error measurement lower than 26%. It is our belief that **this model is a true novelty and a major contribution** to the area due to its adapting characteristics:

- It mines both the periodicity and seasonality of the passenger demand, updating itself regularly;
- It simultaneously uses long-term, mid-term and short term historical data as a learning base;
- It takes advantage of the ubiquitous characteristics of a taxi network, assembling the experience and the knowledge of all vehicles/drivers while they usually use just their own;
- It covers the short-term demand estimation in absolute terms (i.e. pick-up quantities) and also on its fare-based types (i.e. revenues);
- These predictions can be done on-demand and not using any pre-defined time spans;

Notwithstanding the promising results obtained on this particular case study, there are still some issues to handle on future research. Porto is an interesting case study. However, it is just a mid-sized city. Consequently, the latency required to let this models compute their predictions properly do not fill their full potential. These models should be tested on other type of case studies with a larger volume of services and relationship between demand and supply (i.e. Scenario 2). Moreover, these framework is still dependent on a comprehensive set of parameters. Some of them suffered a tuning stage before using - however, others did not (i.e. ma, mi, μ). The fare-based p.d.f. also depends on an user-defined rule set. It is important to develop automatic frameworks to overcome such limitations in the near future. Such topics still comprise open research questions.

Chapter 7

Time-Evolving O-D Matrix Estimation

Nowadays, there is a wide range of ITS applications that are being developed to improve the Urban Mobility. Such improvements are focused on three distinct dimensions: infrastructural, resource usage and passenger oriented. GPS traces are one of the most powerful tools on this research area (independently of their source) by providing a real-time monitoring framework of such mobility. TTP is a relevant problem in many research areas. Throughout Chapter 2 it was possible to observe that it is also transversal to many of the problems addressed in this thesis.

A review on the State-of-the-Art on TTP was already introduced in Section 2.1.3. On the context of Operational control on Taxi Networks, TTP is highly relevant on two decision stages: (i) service selection and (ii) passenger finding. The (i) first one is related with taking/not taking a given service based on its destination. This destination may force to an undesired large vacant cruise time on the return trip due to its running distance or to the poor traffic conditions on a particular pair of road/timestamp. The last one is related on how much time it will take to get to a given urban area/taxi stand where there are favorable service demand conditions (e.g. high service demand in terms of passenger quantity or revenue-based). This Chapter addresses this last problem. By doing so, we expect to meet all the estimations needed to perform a real-time recommendation on the most profitable stand/area to head to in each moment in order to pick-up the next passenger (already referred in Section 2.2.2): the number of services to be demanded in such stand/area (addressed in Chapter 6), the profitability of the services to be demanded in such stand/area (also addressed in Chapter 6) and the travel time needed to go from my current point to that area.

The **Origin-Destination** (O-D) matrix is a State-of-the-Art technique to analyze urban mobility in TN [Lee *et al.*, 2008; Yue *et al.*, 2009; Phithakkitnukoon *et al.*, 2010]. It consists of dividing an urban area into two finite sets of k_o, k_d non-overlapping subregions which entirely cover the initial one. Then, each cell of a $(j_o \times j_d)$: $j_o \leq k_o \wedge j_d \leq k_d$ matrix is used to generate relevant information on the city dynamics, including traffic flow analysis and transportation supply/demand prediction, among others. This information is often inferred using a broad range of algorithms and statistical models over the GPS data streams produced by each network's vehicle. Commonly, an O-D matrix comprises a time-dimension in its cells. Consequently, it is a **discretization** method for both time and space. Despite the continuous characteristics of the GPS streams, most works on O-D matrices based on taxi GPS traces employ batch learning methods [Lee *et al.*, 2008; Liu *et al.*, 2009; Yue *et al.*, 2009; Phithakkitnukoon *et al.*, 2010; Zhang *et al.*, 2011; Qi *et al.*, 2011] which are unable to adapt themselves to sudden drifts on the network status.

The previous Chapter introduced a demand prediction model based on taxi GPS traces. Such demand was decomposed into two axis: the (1) pick-up quantity and (2) the type of service demanded (i.e. long or short connections). While the first axis regards only the number of services demanded on each taxi stand, the second one aims to **typify** such service. The two most important variables on such typification are the **service distance** and its **travel time** (as suggested by the model to compute the service's revenues presented in Section 6.1.1). The approach made to this problem is quite simplistic as we are not interested into determining the exact future revenue value but just a fare-based category (which has a much more narrower domain). But could it be fully or partially applied to Travel Time Prediction (TTP)?

Section 6.5 argued that such model is applicable to any demand prediction problem, independently on its spatial discretization (i.e. stop/stand or city area). Basically, it relies on time series describing what happens on a given area. However, the construction of such series require to maintain a static definition on the spatial discretization level (which is referred by Castro *et al.* [2013] as the most popular approach to this problem). One of the most important issues on building O-D matrices automatically is the penetration rate (i.e. the quantity of ground truth information about the urban mobility on absolute terms) [Tucker, 2009]. The inclusion of multiple data sources to compute such matrix (e.g. bus, taxis, smartphones) can increase this rate. Consequently, this demand predictive model is not a reasonable approach to this problem as it cannot adequately combine such distinct granularities of information.

This Chapter proposes **incremental** discretization techniques to maintain accurate statistics of interest over a **time-evolving** O-D matrix. These statistics can be used as a bedrock for real-time analysis on human mobility dynamics, or as a valuable training input for machine learning algorithms. TTP was selected as a demonstrative application case of this framework using trip-based taxi data. The main contribution of this framework is its applicability to the online estimation of any urban mobility variable of interest using one or multiple data sources, independently of their spatial and/or temporal granularities.

The remainder of this Chapter is structured as follows: Section 7.1 defines the problem while Section 7.2 describes the Case Study addressed in this work along with some details about the data employed in the experiments. The third Section describes the two-layer framework employed to incrementally estimate the O-D matrix. Section 7.4 starts by describing a histogram-based technique to discretize the target variable; then, a discussion is provided on how the histograms can *follow* the evolution of the O-D matrix. A multidimensional discretization model is also proposed to handle discretization in multiple dimensions. Section 7.5 starts by presenting an application case for the methodology (i.e. TTE), along with the experimental setup and its results. Section 7.6 discusses the results obtained, as well as the application of this framework in real-world problems. The related work is briefly revised in Section 7.7. Finally, conclusions are drawn, as well as future research directions on this topic.

7.1 Problem Statement

O-D matrices are a widespread analysis technique employed in many research fields. This work addresses both the generation and maintenance of O-D matrices by mining (A) a *high-speed* continuous flow of origin/destination spatial points (discarding the path followed between the points). This task can be divided into two distinct stages. Firstly, (B) the urban area is divided into two finite sets of non-overlapping k_o, k_d subregions. Then, (C) the origin and destination (i.e. $j_o, j_d : j_o \leq k_o \wedge j_d \leq k_d$) subregions of those initial decomposition are selected as Regions of Interest (ROI) to form the final O-D matrix. A ROI corresponds to an O-D *hotspot* in a city. These problems are formulated along this Section. The symbols and notations used in this Chapter are provided in Table 7.1.

7.1.1 Learning from High Speed Data Streams

Typically, **data streams** comprise a (a) *neverending* flow of data samples. Moreover, the (b) data distribution may not be *stationary*. These characteristics disable the use of many State-of-the-Art ML algorithms. **High-speed** data streams assume that it is not possible to *scan* all the past samples before predicting the target value of the following sample [Gama, 2010]. Let $X = \{x_1, x_2, ..., x_n\}$ be a dataset produced by a high-speed data stream until time instant t. Let $learner()$ be a batch learning algorithm of interest where $model(X, t)$ is the predictive model inferred by it at instant t. Finally, let λ_X be the expected sample arrival rate. The worst-case time complexity of the $learner()$ is, at best, a single-scan complexity (i.e. $O(n)$). (c) High-speed data streams assume the validity of the following equation

$$T(n) = c \times n : \lim_{n \to \infty} \frac{\lambda_X}{T(n)} = 0 \tag{7.1}$$

where $T(n)$ is the time required by the learner algorithm to perform an individual scan for every past n samples, and c is the constant time required to process each sample. In fact, the average number of samples that may be used by any learning algorithm applicable to X is given by $\tau = \frac{c}{\lambda_X}$. In these conditions, a learner is allowed to *inspect* just a *small* number of past samples to update its model before the following sample arrives. In extreme scenarios, the learner may be forced to process just one instance at a time (i.e. $\tau = 1$). This is called an *incremental* learning method. This work follows two assumptions: (1) a GPS data source is an (a) *infinite* stream of (b) *time-evolving* data; (2) its (c) *high* arrival rate implies processing one instance at time.

7.1.2 City Decomposition

A city region is a continuous two-dimensional area (i.e. a subset of $\mathbb{R}^2$), which is difficult to work with. Consequently, it is common practice to *decompose* the city into k disjoint areas to perform any data analysis of interest [Castro *et al.*, 2013]. Let $v_a(lat_a, lon_a)$ be a pair of geographic coordinates representing a *location*. Let $\mathbb{D} \subseteq \mathbb{R}^2$ be an urban area of interest defined by two rectangular vertices with the coordinates $(v_1, v_2) : lat_1 > lat_2 \wedge lon_1 < lon_2$. Implicitly, it is possible to infer the following

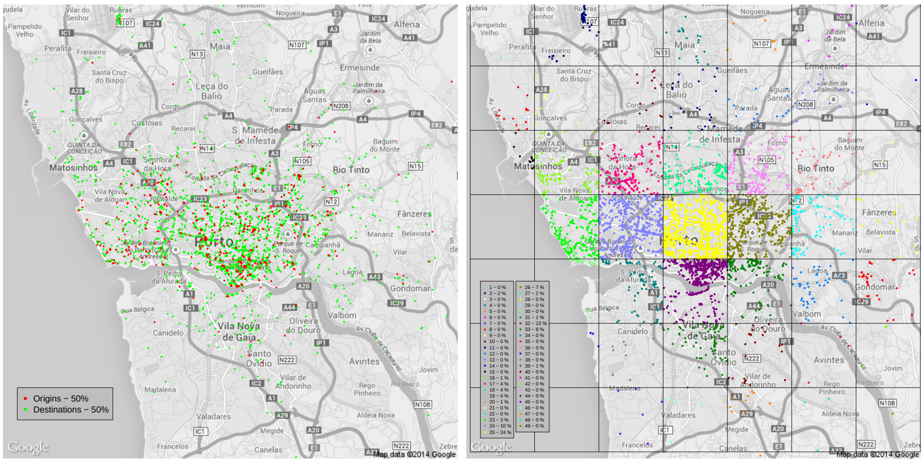
$$\mathbb{D} = [lon_1, lon_2] \times [lat_2, lat_1] \tag{7.2}$$

The **city decomposition** is a pair (Ψ, ψ), where Ψ is a finite set of regions and $\psi : \mathbb{D} \rightarrow \Psi$ is a membership function mapping any location $v_a \in \mathbb{D}$ to a region given by $\psi(v_a) \in \Psi$. This work uses the definitions in the eq. 7.3 presented below. An example of this process is illustrated in Fig. 7.1.

$$\bigcup_{i=1}^{k} \Psi_i = \mathbb{D} \wedge \Psi_i \cap \Psi_l = \emptyset \,, \forall i, l \in \{1, ..., k\} : i \neq l \tag{7.3}$$

Table 7.1: Notation and symbols employed along this Section.

$\mathbb{D}$	urban area to decompose
$v(lat, lon)$	an O-D location represented by a pair of coordinates
Ψ	set of initial subregions / stage1 city decomposition
ψ	membership function to get the location's region in Ψ
k	number of subregions after the stage1 city decomposition
Γ	Parameter set to refine each subregion by density (stage2)
γ_i	i_{th} member of Γ
Ω	set of final subregions / stage2 city decomposition
ω	membership function to get the location's region in Ω
j	number of subregions after the stage2 city decomposition
M	resulting O-D matrix
S	initial *finite* dataset of O-D locations
s_i	data points inside the region Psi_i or Ω_i
$hDim_i$	dimension chosen to split a subregion Psi_i or Ω_i (i.e. lat./lon.)
θ_i	break point to split a subregion in the $hDim_i$
C	Regions of Psi_i that must be refined (i.e. *candidates*)
c	number of regions in C
κ	maximum number of points in memory about one region
s_i	set of data points inside the region Psi_i or Ω_i kept in memory
n	total number of data points/locations in memory
N	total number of data points/locations processed
α	max. threshold for the mass ratio contained in a single O-D region
rt	min. threshold for excessive mass ratio to refine a O-D region
ξ	min. threshold for mass ratio contained in a O-D region
ϕ	min. threshold for mass density in a O-D region
p	split/merging test periodicity on the `layer-on`
ρ_i	mass density of a region Ψ_i
a_i	area occupied by a region Ψ_i
sm_i	set of data points in region Ψ_i
su_i	number of data points contained in a region Ψ_i after its last update
ϑ	highest mass value contained inside one subregion
θ_{Ψ_i}	split point to divide a region Ψ_i into two with equal masses



(a) 5000 O-D pairs' coordinates.

(b) Grid-Based City Decomposition.

Figure 7.1: A naive example on City Decomposition.

7.1.3 ROI selection

The ROI selection is commonly made by employing a threshold-based 0-1 function ω over some user-defined continuous criteria γ_i, such as the O-D location number or *density* within an input subregion Ψ_i. Formally, it is possible to define $\omega : \Psi \to \Omega$ as a membership function $\omega(\Gamma)$, which can be used to iteratively form the ROI set Ω from the original subregion set Ψ. It does so based on the criteria set $\Gamma = \bigcup_{i=1}^{k} \gamma_i$. Consequently, $k \equiv |\Psi| \wedge j \equiv |\Theta|$: $j \leq k \wedge \Theta \subseteq \Psi$.

In various works, only one spatial dimension is considered as they decompose the city according to the destinations or the origins, and not based on the relationship between these locations (e.g., the passenger demand [Lee *et al.*, 2008] or the service offer quantity analysis [Phithakkitnukoon *et al.*, 2010]). An O-D matrix M comprises the relationships between two ROI sets (i.e. origin and destination). It can be formed using two distinct approaches: (i) a unique pair of functions (ψ, ω) to generate both the O-D ROI sets (Ω_o, Ω_d) or (ii) two distinct pairs of functions $\{(\psi_o, \omega_o), (\psi_d, \omega_d)\}$ that produce two separate decompositions on the discretization of the origin/destination continuous spaces. In large TNs, it is expected that $\Omega_o \simeq \Omega_d$ as they contain the city's ROI. However, it is very common to observe *seasonal* changes throughout *time* (i.e.: similarly to human behavior). Therefore, it is common to employ a type-i approach where $\Omega \equiv \Omega_o \equiv \Omega_d$. Consequently, M is represented as a quadratic matrix with size $j_o \times j_d : j_o = j_d$. The temporal discretization is then performed on the matrix cells (as suggested by previous works on related topics [Lee *et al.*, 2008; Yue *et al.*, 2009; Phithakkitnukoon *et al.*, 2010]). The present work follows a type-i approach which also benefits from those assumptions.

7.2 Data Preparation

The case study is the same of the dataset presented in Section 6.1 and so is the data preprocessing. The data was gathered through a non-stop period of nine

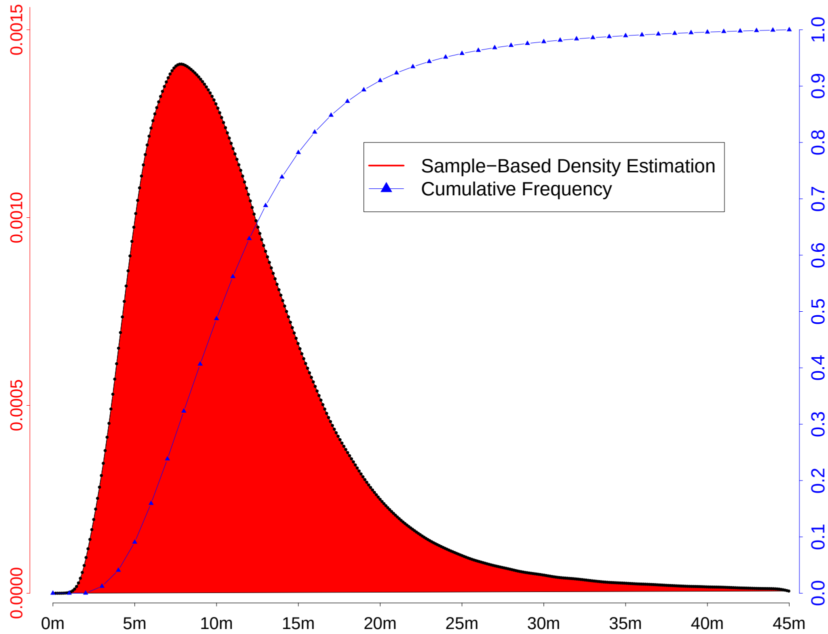


Figure 7.2: Kernel Density Estimation (Gaussian) of the Travel Time (in minutes). Note the *lognormal* form and the low density values (< 0.0014).

months between August 2011 and April 2012. However, in this study the data is organized by trips (similarly to the problem introduced in Chapter 4). Each processed data chunk contains the following seven attributes: the driver's ID, a Julian timestamp, the taxi status (zero/one for vacant/busy), the information about whether the data record concerns the origin or with the destination of the trip, the trip ID and the latitude/longitude coordinates.

The variable of interest in this study is the **travel time** between two O-D locations. This data stream contains two million O-D locations. They correspond to one million taxi trips performed during this period. Fig. 7.2 represents a sample-based estimation of the p.d.f.. The *lognormal* form indicates that the taxi services in the city are usually *short* timed (such as $50\% < 10m$). However, at this granularity, it is not possible to infer more than this as this density illustration chart concerns the trips from several O-D pairs simultaneously.

7.3 Online O-D Matrix Estimation

One of the major problems of decomposing an area into a set of subregions Ψ is guaranteeing that each subregion contains sufficient data points to characterize it. An example of this problem is the popular grid-based decomposition (see Fig. 7.1b), where the city is decomposed on equal-sized regions based on a user-defined width/height [Castro *et al.*, 2013]. Its popularity resides on its simplicity. However, it is naive as it is independent from the data spatial distribution. It results in regions containing an excess/deficit of data samples. The goal with this work is to decompose a city area into equal-sized subregions regarding the *number of points* within (i.e. **mass**).

Let $S = \{v_1, v_2, ..., v_n\} : S \subseteq \mathbb{D}$ be a set of n O-D locations of interest and `spcls` be a *data driven* spatial discretization function defined as follows

$$\texttt{spcls}(\mathbb{D}, S, \Gamma) = \{\Psi, \psi, \Omega, \omega\} : \gamma_i = \gamma_l, \forall i, l \in \{1, ..., k\} \tag{7.4}$$

Finally, let $s_i \subseteq S$ be the set of data points contained in a subregion Ψ_i, where $|s_i|$ is the region mass. The high-level goal is to build an online unsupervised learning method `spcls` that minimizes the value of mass standard deviation ($\sigma_{|s_i|}$).

The incremental estimation of an O-D Matrix without any prior knowledge is a difficult task. A two-layer discretization algorithm is proposed to overcome this problem. In the `layer-off`, (A) a batch learning algorithm starts by performing hierarchical mass-based clustering [Ting and Wells, 2010] to find the best k subregions that meet this last high-level goal. Then, a density-based function is defined as ω. Finally, the O-D matrix is built based on the resulting Ω. The second layer (`layer-on`) (B) goes from the output of the previous layer to incrementally update a *sufficient* amount of statistics about the regions in Ω. This methodology is thoroughly described along the next Section.

7.3.1 `layer-off`: Batch O-D matrix Estimation

Let Ψ_i be a *rectangular* subregion defined by two vertices $v_{i,1}, v_{i,2}$ whose coordinates are defined as follows:

$$lat_{i,1} = \max(lat_i), lon_{i,1} = \min(lon_i),$$
$$lat_{i,2} = \min(lat_i), lon_{i,2} = \max(lon_i) : lat_i, lon_i \in \Psi_i \tag{7.5}$$

This algorithm starts by initializing $\Psi = \Psi_1, k = 1 : \Psi_1 = \mathbb{D}$. Then, it iteratively runs a cycle composed of five steps: firstly, it selects the i_{th} subregion as $\arg\max_{i \in \{1,...,k\}} |\Psi_i|$. Secondly, the length of the vertical/horizontal i_{th} subregion is computed using the *Haversine* distance between two geographic coordinates [Robusto, 1957]. Then, one of the latitude/longitude is selected as the largest/shortest dimension $hDim_i, lDim_i$ based on that length. The third step consists on finding the **binary** split point θ_Ψ which divides the region space Ψ into two regions with *equal* masses. Fourthly, it creates a new $k + 1_{th}$ subregion defined by $\{\Psi_{k+1}, s_{k+1}\}$, where $\Psi_{k+1} \subset \Psi_i$ is defined by the area's breakpoint θ_Ψ of the $hDim_i$ dimension. s_{k+1} is defined as follows:

$$s_{k+1} = \Big\{ v_o |\, v_o[hDim_i] \geq \theta_\Psi, \forall o \in \{1, ..., |s_i|\} \Big\} \tag{7.6}$$

Finally, the algorithm updates the k number of partitions as $k' = k + 1$, as well as the sets Ψ_i, s_i as Ψ'_i, s'_i defined in the following equations (where $\Psi_{k'}$ stands for the latest created subregion).

$$\Psi'_i = \{v_o |\, v_o \in \Psi_i \wedge v_o \notin \Psi_{k'}, \forall o\} \tag{7.7}$$

$$s'_i = \{v_q | v_q \in s_i \wedge v_q \notin s_{k'}, \forall q\} \tag{7.8}$$

This cycle only stops when $\vartheta \leq \alpha$, where α is a user-defined parameter (commonly a small ratio of n) and $\vartheta = \max_{i \in \{1,...,k\}} |s_i|$. It defines the desired *granularity* level.

The suggested mass-based partitioning method follows closely the method proposed by Ting and Wells [2010]. This application case is a two-dimensional case as $\mathbb{D} \subseteq \mathbb{R}^2$. Its implementation is also made through a **Half-Space tree** where the concept of *space* is given by each region's **mass**. The split point θ is computed as the *median* value of $hDim_i$ on s_i. Consequently, ψ will be a decision tree where the *leaves* will contain a subregion and the *nodes* will contain a split-point condition regarding one of the two spatial dimensions.

After the initial decomposition, a ROI selection is performed. Let ρ_i be the mass density of a region Ψ_i given by $\rho_i = |s_i|/a_i, \forall i$, where a_i is an area occupied by the region Φ_i. Let ϕ, rt be a user-defined minimum density-based threshold and a mass-based threshold ratio, respectively. Let ξ denotes a minimum mass-based threshold ratio where $\xi \ll \alpha$. The membership function $\omega : \Psi \to \Omega_1$ can be defined as follows:

$$\omega_1(\rho_i, \phi) = \begin{cases} 1 \; if \; \rho_i \geq \phi \vee |s_i| \geq \frac{\alpha \times n}{1+rt} \\ 0 \; if \; \rho_i < \phi \wedge |s_i| < \frac{\alpha \times n}{1+rt} \end{cases} : 0 < rt \ll 1 \tag{7.9}$$

The remaining regions form a set of c region *candidates* $C = \{\Psi_i | \Psi_i \in \Psi \wedge \psi \notin \Omega_1\}$ which may need to be *refined*. The goal now is to find subregions in each region C_i which have, at least, $1 - rt$ percentage of the total data points $\in C_i$, i.e. $|s_i|$. For that, the method runs a four-step cycle: firstly, it selects the i_{th} subregion candidate as $\arg\min_{i \in \{1,\ldots,c\}} \rho_i$. Secondly, it discards the candidate C_i if $|s_i| < \xi \times n$. Such test aims to *filter* regions without a relevant quantity of O-D flows within. Thirdly, it finds a split point θ to divide C_i into $\{C_{c+1}, C_{c+2}\}$ as $|s_{c+1}|/|s_{c+2}| \simeq rt$, using an approach similar to the one employed in stage 1. Finally, C and Ψ are updated as follows $C' = C \setminus C_i$ and $\Psi' = \Psi \setminus \{C_i\} \cup \{C_{c+1}\} \cup \{C_{c+2}\}$. The ROI set Ω is updated as $\Omega' = \Omega \cup \{C_{c+2}\}$ if $\omega(\rho_{c+2}, \phi) = 1$. Otherwise, C_{c+2} returns to the candidate set as $C'' = C' \cup \{C_{c+2}\}$. This cycle runs continuous until $C \equiv \emptyset$.

7.3.2 `layer-on`: Incremental O-D matrix Estimation

Let $S_t = \{v_1, v_2, ...\}$ be an **infinite** set of locations where N is the number of samples achieved at time instant t defined as $|S_t| = N : \lim_{t\to\infty} N = \infty$. Let $sm = \{sm_1, ..., sm_k\}$ be the set containing the set of data points sm_i within a subregion Ψ_i and n be the number of points stored in *memory* at instant t. After performing the *first* run of `layer-off`, $n = N$ and $s \equiv sm$. However, this relationship cannot be maintained as the *memory* has a bounded domain, while N has an unbounded domain. Therefore, n is constrained as $\lim_{t\to\infty} n \ll N$.

To define the domain boundaries of n, it is necessary to describe the *minimum* amount of information required to characterize the spatial data distribution in Ψ. This information can be used to reconstruct Ψ, Ω at all times by using the points in $sm : \sum_{i=1}^{k} |sm_i| = n$. To do so, the `layer-on` starts by setting the maximum number of points $\kappa = \arg\max_{i \in \{1..k\}} |sm_i|$ as the one obtained at the time instant immediately after the *first* run of `layer-off`. Consequently, the domain of n meets its constraint as $\lim_{t\to\infty} n = \kappa \times k \ll N$.

Let $\bar{lat}_i, \bar{lon}_i$ be the average latitude/longitude of the $|s_i|$ O-D points in region Ψ_i at time instant t. This algorithm iteratively processes each new sample $v_N \in S_t$ in a three-step loop: firstly, it determines $R_i = \psi(v_N) : R_i \in \Psi$ as the O-D subregion to which v_N belongs. Secondly, it updates the number of points $|s_i|$, as well as $\bar{lat}_i, \bar{lon}_i$, based on v_N. sm_i is also updated as

$sm'_i = sm_i \cup \{v_N\}$. However, if $|sm'_i| > \kappa$, a *forgetting* mechanism is launched. The algorithm deletes the most outdated data point (sm'_{i1}) from the memory as $sm''_i = sm'_i \setminus \{sm'_{i1}\}$. Its goal is to maintain the n inside a bounded domain. Finally, the algorithm determines which of the current partitions in Ψ meets the **merge/split** criteria. This operation is periodically performed, where p represents its period. p can be defined in time (i.e. $p \geq \lambda_{St}$, where λ_{St} stands for the expected arrival rate of new locations $v_N \in S_t$) or in space (i.e. each p samples) as $p \geq 1$. The value of p sets how *reactive* our model will be.

Merging Partitions

By merging, the algorithm aims at *recovering* the regions from the ROI set Ω where the number of O-D points increases more than expected. Let su represent the region mass after its last update (i.e. merge/split). su is initialized as $su_i = |sm_i|$: $i \in \{1, ..., k\}$ right after the *last* run of the `layer-off`. The `merge` operator is launched in every region in $C = \{C_i | C_i \notin \Omega \wedge C_i \in \Psi \wedge |s_i| > 2 \times su_i\}$. The merge operation starts by finding the deepest conditional *node* of ψ which divides C_i from another region Ψ_{old} (it is an operation with a worst-case time complexity of $O(k)$). Secondly, the operation transforms the *node* into a leaf node with the cluster of the newest region Ψ_{new} defined as $\Psi_{new} = \Psi_{old} \cup C_i$. Ψ, k and s are updated accordingly as $\Psi' = \Psi \cup \{\Psi_{new}\} \setminus \{\Psi_{old}\} \setminus \{C_i\}$, $k' = k + 1$ and $s' = s \cup \{s_{new}\} \setminus \{s_{old}\} \setminus \{s_i\}$. Ω and su are also updated as $\Psi_{new} \in \Omega$: $|s_{new}| \geq \xi \times N$ and $su_{new} = |s_{new}|$.

Splitting Partitions

The splits follow a similar approach as the one proposed in Stage 1 of the `layer-off`. The `split` operator is triggered in every region in $C = \{C_i | C_i \in C : C_i \in \Omega \wedge |s_i| > \alpha \times N\}$. The main difference resides in defining the split point θ. In this layer, it is not possible to conduct a single-scan operation on multiple data points $\in sm_i$ to calculate the median. Instead, $\bar{lat}_i, \bar{lon}_i$ are used, - which are easily maintained following an *incremental* logic.

7.3.3 Two-layer framework

Similarly to many incremental learning algorithms [Gama and Pinto, 2006], the `spcls`$(\mathbb{D}, S, \Gamma)$ maintains two distinct *layers*: the `layer-off`, which determines the best possible ROI set Ω by employing unsupervised batch learning methods over the entire dataset available, and `layer-on`, which approximates Ω by updating itself to each new data point. This flexibility comprises an error which grows as the `split` operator is invoked in the `layer-on`. To mitigate this effect, the framework can launch the `layer-off` on-demand. The foundation for this ability is s. It is a set of data points that keeps the most recent data points of each existing region Ψ_i. s is maintained using a *sliding window* whose size is determined by the constant κ, which is obviously correlated to the parameters α and n.

Therefore, the `spcls` can be classified as an unsupervised learning method which is also incremental. Its parameter set Γ is defined as $\Gamma = \{n, \alpha, \phi, rt, \xi, p\}$. The most sensitive parameters are ϕ and ξ as they define the boundaries of Ω. rt

just defines if a region may be refined or not. n and α affect spatial complexity, while p causes small drifts on the time complexity.

The pseudo code of this two-stage partitioning is displayed in Fig. 7.3 and Fig. 7.4. The O-D matrix is formed as $M[r, i]$ denotes a cell containing information on the mobility flows from the region Ω_r to the region Ω_i. The matrix evolution over time poses constrains when storing this information, because not only should it be maintained incrementally, but it should also be easily *decomposed* in order to follow the splits/merges performed. Incremental Histograms are proposed to meet these constraints, which are described in the following Section.

function STAGE1-CITY-DECOMP($\mathbb{D}, S, n, \alpha$)
 $s[1] \leftarrow S; \Psi[1,] \leftarrow \mathbb{D}; np[1], \vartheta \leftarrow n;$
 $\psi \leftarrow$ `leaf`$(1); k, i \leftarrow 1;$
 while $(\vartheta > (\alpha \times n))$ **do**
  $len_{lat} \leftarrow$ `haversineDist`$(\Psi[i][1,1], \Psi[i][2,1]);$
  $len_{lon} \leftarrow$ `haversineDist`$(\Psi[i][1,2], \Psi[i][2,2]);$
  $hdim \leftarrow 1;\ \ k \leftarrow k+1;$
  If $(len_{lon} > len_{lat})$ **then**
   $hdim \leftarrow 2;$
  $\theta_\Psi \leftarrow$ `median`$(s[i][, hdim]);$
  $s[k], np[k] \leftarrow$ `getDataPoints`$(hdim, \theta_\Psi, s[i]);$
  $s[i] \leftarrow$ `getDisjointDataPoints`$(s[i], s[k]);$
  $\psi \leftarrow$ `updateLeafToCondition`$(i, k, hdim, \theta_\Psi);$
  $np[i] \leftarrow |np[i] - np[k]|;$
  $i, \vartheta \leftarrow$ `maxMassCluster`$(np);$
 end while
 $\Psi \leftarrow$ `runTree`$(\mathbb{D}, \psi)$; **return** $\{\Psi, \psi, s, k, np\};$
end function

Figure 7.3: Stage 1 City Decomposition.

7.4 Incremental Data Discretization using Histograms

Histograms are a State-of-the-Art method in exploratory data analysis. They make it possible to discretize continuous variables into *intervals*. This approach is a common building block of many machine learning algorithms (e.g. Bayesian Learning [Domingos and Pazzani, 1997]), and for that reason it is proposed as a tool to maintain accurate statistics over a time-evolving O-D matrix.

Let H be defined as the set of all histograms in M (i.e. the histograms describing a variable of interest in each cell of M). Let $h_{o,d} \in H$ represent a histogram of q intervals discretizing a continuous spatiotemporal variable of interest $X_{o,d} = \{(x_i, v_i) | v_i \in \Psi_o \forall i\}$ and $|h_{o,d}|$ denotes the mass within. $X_{o,d}$ describes directional interactions between the O-D regions $\Psi_o, \Psi_d \in \Omega$ (e.g.: x_i may represent a value of any variable of interest). $h_{o,d} = (B, F)$ can be defined as a set of breakpoints $B = \{b_1, ..., b_{q-1}\}$ and a set of frequency counts $F = \{f_1, ..., f_q\}$. This Section describes a fully incremental strategy to maintain

```
function STAGE2-DNSTY-REFIN(Ψ, ψ, s, k, np, α, υ, rt)
    c ← 0; j ← 0;
    for i ∈ {1, ..., k} do
        ρ[i] ← np[i]/getArea(Ψ[i]);
        if (ω1(ρ[i], υ) = 1) then
            j ← j + 1; Ω[j, ] ← Ψ[i];
        else
            c ← c + 1; C[c, ] ← Ψ[i];
        end if
    end for
    while c > 0 do
        i, j ← minDensityCluster(Ψ, C, s);
        if (np[i] < n × ξ) then
            C, c ← removeRegionFromCandidates(C, C[j, ], c);
            Continue;
        end if
        len_lat ← haversineDist(Ψ[i][1, 1], Ψ[i][2, 1]);
        len_lon ← haversineDist(Ψ[i][1, 2], Ψ[i][2, 2]);
        hdim ← 1;  c ← c + 1;
        If (len_lon > len_lat) then
            hdim ← 2;
        θ_C ← getSplitPoint(s[i][, hdim], rt);
        k ← k + 1; c ← c + 1;
        s[k], np[k] ← getDataPoints(hdim, θ_C, s[i]);
        s[i] ← getDisjointDataPoints(s[i], s[k]);
        ψ ← updateLeafToCondition(i, k, hdim, θ_C);
        np[i] ← |np[i] − np[k]|;
        Ψ[k] ← Ψ[i]; Ψ[i] ← ψ(Ψ[i]); Ψ[k] ← ψ(Ψ[k]);
        C, c ← removeRegionFromCandidates(C, C[j, ], c);
        ρ[k] ← np[k]/getArea(Ψ_k);
        if (ω(ρ[k], υ) = 1) then
            j ← j + 1; Ω[j, ] ← Ψ_k;
        else
            c ← c + 1; C[c, ] ← Ψ_k;
        end if
    end while
return {Ψ, ψ, k, Ω, j, s, np};
end function
```

Figure 7.4: Stage 2 Density-based ROI Selection.

histograms on $X_{o,d}$ in real-time on distinct dimensional levels.

7.4.1 The Partition Incremental Discretization `PiD`

The `PiD` is a fully incremental algorithm capable of maintaining accurate histograms of never-ending streams of data [Gama and Pinto, 2006]. We propose the employment of the algorithm to maintain histograms of *equal width*, such as $(b_i - b_{i-1}) = (b_l - b_{l-1}) = \delta_q$, $\forall i, l$.

This algorithm works on two different layers. Let q, q_1 be two user-defined number of bins and $[\nu_1 : \nu_2]$ be the range of $X_{o,d}$. q stands for the *desired* number of bins, while q_1 is used as input parameter to the `layer1` defined as $q_1 \gg q$. The `layer1` is initialized as $F = \{f_i | f_i = 0, \forall i\}$ and $B = \{\nu_1, ..., \nu_2\}$: $(b_i - b_{i-1}) = \delta_{q_1}, \forall i$. Then, the algorithm runs continuously, incrementing f_i every time a sample (x_a, v_a) is added, where $v_a \in \Psi_o$. If $x_a < \nu_1 \vee x_a \geq \nu_2$, a new bin is added to such extremity with the step δ_{q_1}. The `split` operator is triggered on a bin if $f_i > \eta$, where η is a user-defined parameter usually defined as a ratio of the histogram mass. Consequently, two bins are created, each one comprising *half* of the interval $[b_i, b_{i+1}]$ and containing the same frequency $f'_i = f_i/2 : f'_i \in \mathbb{N}$.

The `layer2` is launched every time the user needs to analyze the data. It iteratively merges the bins in `layer1` to meet the desired q in terms of size intervals δ_q. The main advantage of maintaining these layers is that it is possible to easily produce histograms of different sizes each time it is necessary to discretize the domain variable. Additional details on the `PiD` algorithm are provided in [Gama and Pinto, 2006].

7.4.2 Following the O-D Matrix Evolution

One of the major issues of building histograms is the definition of q. There is not a well-established general strategy to do so. Different strategies may be employed depending on the user's purposes. The main contribution of `PiD` is that q does not need to be constant: it can be either time or sample dependent. Whenever Ψ, Ω and M change over time, H must follow the merge and split operations. Let δ_{min} be the minimum interval size in H. The interval widths in H must be subjected to the following constraint:

$$H = \{h_i(B, F) | \exists\, a \in \mathbb{N} : (b_l - b_{l-1}) = \delta_{min} \times 2^{a-1}, \forall i, l\} \tag{7.10}$$

Consequently, the problem of merging two histograms $h_{o_1,d}, h_{o_2,d}$ into a single one $h_{o,d}$ can be defined as

$$q_{o,d} = \frac{\max b_l - \min b_l}{\delta_{o,d}} : b_l \in \{B_{o_1,d} \cup B_{o_2,d}\} \tag{7.11}$$

where $\delta_{o,d} = \max(\delta_{o_1,d}, \delta_{o_2,d})$. Then, the `layer2` is employed to turn the histograms $h_{o_1,d}, h_{o_2,d}$ into equal-width histograms as $q_{o_1,d} \equiv q_{o_2,d} \equiv q_{o,d}$. Finally, the frequency set is defined as $F_{o,d} = \{f_i | f_i = f_{io1,d} + f_{io2,d}, \forall i \in \{1, ..., q_{o,d}\}\}$.

The constraint defined in eq. 7.10 makes the `layer2` task *easier* by guaranteeing that $\delta_i \mod \delta_{min} = 0, \forall i$. This property guarantees that all the histograms in H are *additive* between each other. The division of $h_{o,d}$ into $h_{o_1,d}, h_{o_2,d}$ is a simple operation where $B_{o_1,d} = B_{o_2,d} = B_{o,d}$ and $F_{o_1,d} = F_{o_2,d} = \{f_i \in \mathbb{N} | f_i \simeq f_{io_d}/2\,, \forall i\}$.

7.4.3 Dimensions and Hierarchies

The histograms are a well-known approach to provide sample-based discrete approximations of a Probability Density Function (p.d.f.) on the value of a continuous variable $X_{o,d}$. However, it is known that the mobility dynamics (such as the number of taxi pick-ups/drop-offs [Yue *et al.*, 2009] in a region, or a bus round-trip time [Matias *et al.*, 2010]), follow a *bimodal* distribution (e.g.

peak/non-peak hour) throughout the day. Mobility dynamics can even be multimodal if a larger time span is considered, such as one week (workday/weekend). This p.d.f. can be difficult to learn online. To overcome this problem, **Dimensional Hierarchies** are proposed as a *flexible* method to discretize other dimensions describing $X_{o,d}$ (e.g. the **temporal**).

Let Z be a set of χ dimensions related to $X_{o,d}$, where $Z_i \in Z$ denotes a hierarchized set of $|Z_i|$ dimensional attributes. Chen *et al.* [2005] firstly propose it as a method to discretize $X_{o,d}$ on multiple χ dimensional axis. Depending on the amount of data available on $X_{o,d}$, the discretization layers on each axis may have different information granularities (i.e. *zoom* values).

This work adapts this definition by redefining Z as a hierarchical set of dimensions. Let $Z^\chi = \bigcup_{i=1}^{\chi} Z_i$ be an **ordered** set of *multidimensional* attributes where the order is user-defined (depending on the purpose of the histogram). The discretization intervals in each zoom level may also be user-defined or data-driven (e.g. breakpoints on average values and/or quartiles). The proposed framework maintains distinct histograms $h_{o,d,i}$ on every zoom level i by continuously running the `layer1` over the histograms. The `layer2` is triggered prior to each statistical analysis of $h_{o,d,i}$ only if $|h_{o,d,i}| > \epsilon_i = 2 \times \epsilon_0$. ϵ_0 denotes a user-defined parameter for the minimum amount of available data points to trigger the `layer2` on the zero-level dimensional hierarchy (i.e. base histogram; without dimensional discretization). An illustrative example of this framework is provided in the Fig. 7.5. In this example, h_1 stands for a histogram of the maximum instant speed of a taxi driven from region o to region d by a 40-year-old female subject between 07am and 11am.

Z^χ establishes relationships between attributes of distinct dimensions. Conversely to Z, initially proposed in [Chen *et al.*, 2005], Z^χ does not allow different levels of discretization in different dimensions. It is necessary to maintain additional histograms if this analysis is intended. This step works as a threshold search for the *nearest neighbor*, which tries to build statistics using past samples where the descriptive variables are similar to the present variables. It does so by maintaining a decision tree of each O-D where the goal is to find the histogram which gives the best approximation to the present scenario. Consequently, the goal is to describe $X_{o,d}$ using multiple attribute-based histograms which are more likely to approximate unimodal p.d.f. (rather than multimodal p.d.f.).

7.5 Experiments

This Section presents the experimental work performed in this context. It starts by describing a *naive* online learning model built over the proposed framework to perform Travel Time Estimation (TTE). Secondly, the experimental setup and the evaluation metrics are described. Finally, the results obtained are presented.

It is important to highlight that we do not want to claim this induction model as a contribution to the TTE problem *per se*. The literature on this topic is extensive [Mendes-Moreira *et al.*, 2012]. The results obtained thorough this model work as a proof of concept on the applicability of this framework to maintain accurate real-time statistics on the TN-based urban dynamics.

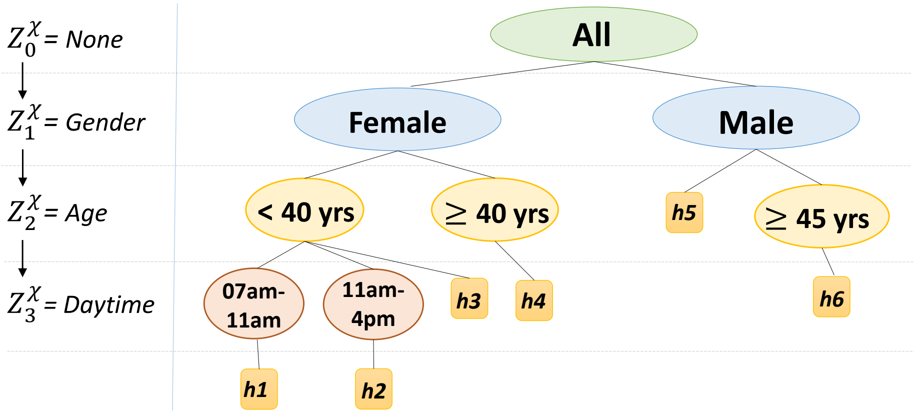


Figure 7.5: Example of a multidimensional hierarchy to discretize attributes. Note that the zoom level and the discretization intervals may not be constant.

7.5.1 An Application for Travel Time Estimation

TTE aims to predict the cruise time of a given trip between an O-D pair of locations. It can be defined as short or long-term depending on the predicting horizons [Mendes-Moreira *et al.*, 2012]. The most common is the short-term one. It is commonly employed in Automatic Traveler Information Systems (ATIS) and Navigational GPS devices [Chien *et al.*, 2002; Carrascal, 2012]. Producing online predictions on this stochastic variable is a difficult problem. Typically, these systems employ batch regression models along with online models (such as time-series analysis and/or state-based induction models) to update the initial predictions using the real-time vehicle trace [Chien *et al.*, 2002; Chen *et al.*, 2004; Bin *et al.*, 2006].

This work considers the TTE in a more classical approach: given a pair of O-D locations (v_o, v_d) at time instant t, the target variable is the cruise time between these locations, expressed as $\beta_{o,d,t}$. Let $h_{o,d,z}$ be the most suitable histogram to describe the present scenario given the values of the dimensional attributes defined in Z^χ. Let $\bar{b_i} = (b_i - b_{i-1})/2$ denote the center of the interval corresponding to the upper and lower bounds on the bin i. $\beta_{o,d,t}$ can be obtained as follows:

$$\beta_{o,d,t} = \frac{1}{\Upsilon} \times \sum_{i=1}^{q_{o,d,z}} \left[\left(\frac{f_{o,d,z,i}}{\max(f_{o,d,z,i})} \right)^2 \times \bar{b}_{o,d,z,i} \right] \tag{7.12}$$

$$\Upsilon = \sum_{i=1}^{q_{o,d,z}} \left(\frac{f_{o,d,z,i}}{\max(f_{o,d,z,i})} \right)^2 \tag{7.13}$$

where T denotes the time elapsed between t and $t-1$. The quadratic normalization of the frequencies in the eq. 7.12 aims at minimizing the well-known vulnerability of equal-width discretization techniques: outliers [Gama and Pinto, 2006].

7.5.2 Experimental Setup

To define $q_{o,d,i}$, it was necessary to have a rule capable of dealing with large volumes of samples as long as it meets the constraint expressed in the eq. 7.11. It can be expressed as follows:

$$q_{o,d,i} \simeq \lceil |h_{o,d,i}|^{(\frac{2}{3})} \rceil : b_{o,d,i} - b_{o,d,i-1} \leq 120 \wedge q_{o,d,i} \in \mathbb{N} \tag{7.14}$$

This rule is merely user defined and it was chose after carrying out a sensitivity analysis on five different equations to set the number of bins $q_{o,d,i}$. These methods were developed following closely the Section 2 in [Birge and Rozenholc, 2006]. A sensitivity analysis was performed for the parameters n, rt, ϕ, ξ and α based on a simplified version of Sequential Monte Carlo method over an older dataset (the reader can consult the survey in [Cappé *et al.*, 2007] to know more about this topic). These parameters were chose because are the ones which variations reflected some changes on the method outcome during such analysis. The tested values were all the *admissible* combinations (i.e. $\alpha > \xi$) on the following ranges: $n = \{2000, 4000, 6000, 10000\}$, $\alpha = \{0.02, 0.05, 0.10\}$, $rt = \{0.1, 0.2, 0.3\}$, $\xi = \{0.005, 0.01\}$ and $\phi = \{0.3, 0.5, 0.8\}$. The best combination of values for this set of parameters was then selected to conduct these experiments. This combinations is is detailed in Table 7.2, along with the remaining ones.

In the experiments, the time dimension is expressed in seconds. They were conducted using the R Software [R Core Team, 2012]. The graphical representations of the city O-D regions were obtained using the package `[RGoogleMaps]`. The `layer-off` was just triggered once to start the algorithm. $Z = \{Distance, Time\}$ were the dimensions considered, while $Z^{\chi} = \{$`haversineDistance`, `dayTime`, `weekType`, `dayType`$\}$ was defined as the multidimensional hierarchy set.

The (1) `haversineDistance` has an unique breakpoint based on historical data (the average distance in the trips described by $h_{o,d,0}$). The remaining three attributes have breakpoints for their intervals defined as (2) {07h-11h, 11h-16h,16h-21h, 21h-07h}, (3) $\{Workday, Weekend\}$ and the (4) seven days of the week, respectively. The sea was considered a constraint to compute a region area. This was done by defining a minimum longitude along the coast. The areas were calculated by approximating the constraints using trapezoids.

The histograms built were used as input for the prediction model presented here to infer the travel times of the most recent 250,000 trip samples (i.e. O-D trip pairs (v_o, v_d)). The attribute values of each sample were used to select the most suitable histogram. The option chosen was to build histograms in every region $\Psi_i \in \Psi$, maintaining a quadratic $k \times k$ O-D matrix over the entire city. However, the histograms were not employed if $\Psi_o \notin \Omega$ in the current time instant. Whenever there is no zero-level histogram available, a naive approach is followed by assuming a constant cruising velocity of 30 km/h. Predictions were also produced on the travel time interval by selecting the minimum number of consecutive bins containing, at least, 75% of the mass $|h_{o,d,i}|$.

To demonstrate robustness, the model was tested in three distinct scenarios: maintaining the histogram framework over an O-D matrix built on a grid-based City Decomposition (by dividing the city into 7×7 equally sized areas) and

Table 7.2: Parameter Setting used in the experiments.

Parameter	Value	Description
n	6000	number of O-D points used on the `layer-off`
α	$0.05 \times N$	maximum mass ratio contained in a O-D subregion
rt	0.1	minimum excessive mass ratio to refine a O-D subregion
ξ	$0.01 \times N$	minimum mass ratio contained in a O-D subregion
ϕ	0.5 of the avg. mass density	minimum mass density in a O-D subregion
p	each sample	split/merging test periodicity on the `layer-on`
$q_{o,d,i}$	eq. 7.14	desired number of bins on `layer2`
q_1	270 : width (δ_{q_1})=10s	initial number of bins in `layer1`
ϵ_0	100	minimum number of samples to build a zero-level histogram
δ_{min}	2s	minimum interval width for the histograms
η	0.30	maximum total mass ratio contained on a single bin in `layer1`

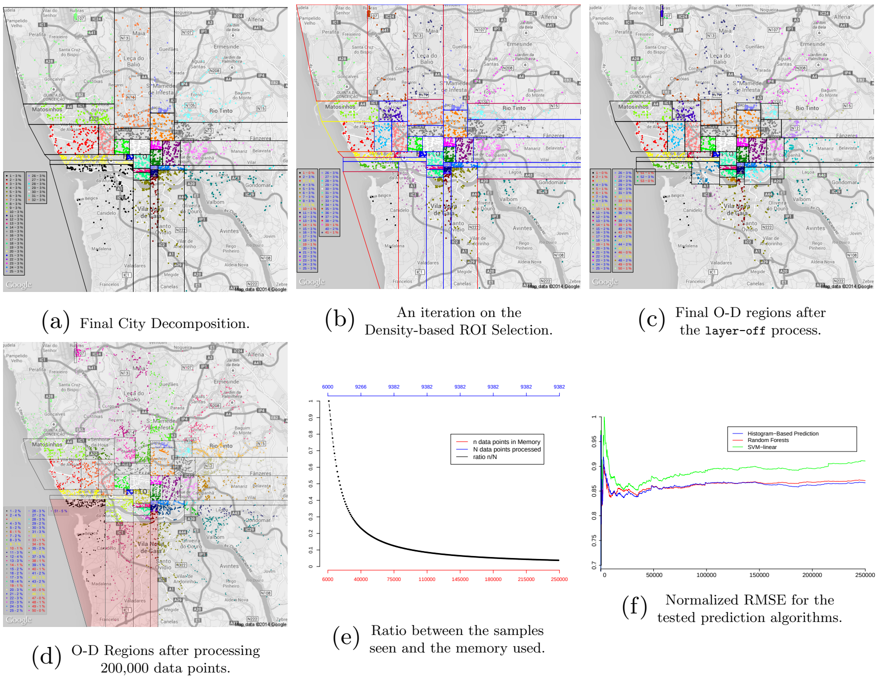
comparing it with the mass-based approach; employing zero-level histograms *vs.* the proposed multidimensional discretization, and monitoring the performance of the induction algorithm over time against two State-of-the-Art offline regression methods on TTP: the Random Forests [Mendes-Moreira *et al.*, 2012] and the Support Vector Machines [Mendes-Moreira *et al.*, 2012; Bin *et al.*, 2006]. The regression features were defined as follows: (1) Day, coded as a sequence of integer numbers; (2) Starting Time (in seconds) and (3) Day of the week. The packages `[randomForest]`, `[e1071]` provided the methods' implementations used in the experiments. They were executed using their default parameter setting. Each O-D pair was treated as an independent regression problem (as in the induction model proposed).

7.5.3 Results

Fig. 7.6 illustrates the multiple stages of estimating the O-D matrix using HS Trees. The first four subfigures report the Offline Estimation process, while the fifth reports a `layer-on` iteration. The fifth subfigure compares the memory used during the online estimation with the number of data points processed. The last subfigure reports the evolution of the algorithms' prediction error throughout time. This report is based on a normalized RMSE. This metric is calculated firstly by computing the average RMSE throughout time for each predictive method. Then, all the series obtained are divided by the same maximum value. The aggregated results for all the tested samples are presented in Tables 7.3 and 7.4. The effects of the multidimensional discretization framework are exemplified in Fig. 7.7. In average, the `layer-off` took 92 sec. of computational time on each run, while the `layer-on` just took 0.01*sec.* per iteration.

7.6 Discussion

Five main conclusions can be drawn from the results presented. The (1) proposed O-D matrix estimation method is able to discover **dense** ROI. Note the evolution from Fig. 7.6a to Fig. 7.6c. The area uncovered in the northwest area is the city's airport. This ROI was initially contained in a vast area, but the



(a) Final City Decomposition.

(b) An iteration on the Density-based ROI Selection.

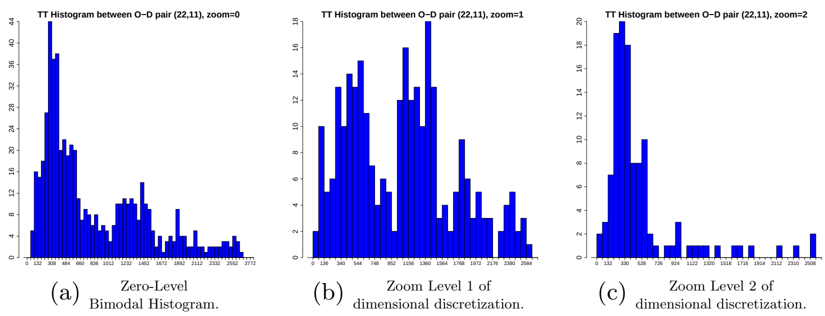
(c) Final O-D regions after the `layer-off` process.

(d) O-D Regions after processing 200,000 data points.

(e) Ratio between the samples seen and the memory used.

(f) Normalized RMSE for the tested prediction algorithms.

Figure 7.6: Illustration of the Time-Evolving OD-Matrix Estimation Process. Note the density refinement in the northwest airport area discovered in C), the ability to adapt to a large increase in the region's mass in (D), and the low memory requirements to maintain a time-evolving framework in (E).



(a) Zero-Level Bimodal Histogram.

(b) Zoom Level 1 of dimensional discretization.

(c) Zoom Level 2 of dimensional discretization.

Figure 7.7: Example of the multidimensional discretization effects of the travel time density function. Note that the zero-level histogram approximates a bimodal p.d.f while the zoom=2 in (C) highlights a unimodal p.d.f. by selecting the trips occurred in 11am-4pm.

density refinement staged uncovered its true shape. However, such refinement is only performed by launching the `layer-off`. This is one of the main drawbacks of this methodology. Setting an adequate periodicity to launch this *layer* prepares the system's ability to react to the formation of highly dense zones. Yet, a high periodicity will largely increase the computational effort in the processing task.

Table 7.3: TTE Prediction Evaluation comparing a Grid-Based City Decomposition and a Mass-Based City Division.

	RMSE	MAE	Average Interval Width	In Interval (%)
Grid-Based	349.33	222.22	466.06	66.54 %
Mass-Based	306.34	198.66	531.47	79.10 %

Table 7.4: Comparison of different online/batch predictive models on TTE.

	RMSE	MAE	Average Interval Width	In Interval (%)
Random Forests	307.94	209.89	*Not Applicable*	
SVM-linear	321.96	**189.87**	*Not Applicable*	
Histogram-Based MaxZoom=0	316.69	210.60	557.38	79.13 %
Histogram-Based MaxZoom=4	**306.34**	**198.66**	**531.47**	**79.10 %**

The system is able to maintain a (2) **flexible O-D matrix** over time by updating the **low** levels of memory required. Fig. 7.6d highlights the framework's flexibility to sudden changes in the cluster's masses. Fig. 7.6e shows that the algorithm maintains a *logarithmic* space complexity. Note that this complexity is not affected by the `layer-off` launching periodicity.

The mass-based city decomposition (3) **outperforms** the grid-based one. It is not only able to discover equal-mass ROI, but also to maintain equally-sized cells on the O-D matrix. It is not surprising to find that the grid-based histograms are less suitable than the histogram proposed in this thesis for TTE (as observed in Table 7.3). The grid-based simplicity is its best quality as well as a strong drawback. The proposed HS trees are also simple but **data driven**, which strengthens the distribution of data in their leaves.

This incremental approach (4) is more **suitable** than the State-of-the-Art batch regression models in the present TTE scenario. Since the models obtained from the training set are not updated using the newly arrived samples, their performance decreases throughout time (see Fig. 7.6f and Table 7.4). Even if the SVM-linear presents the lower MAE in Table 7.4, it is highly questionable to claim that it would be able to maintain such performance, especially if we see its evolution in Fig. 7.6f. The mean deviation (i.e. $\simeq$ 200sec.) also reflects the stochasticity of the variable, demonstrated in Fig. 7.2.

It is also important to highlight the histogram's ability to produce accurate **intervals** in the domain of the target variable. The accuracy of these intervals can be partially user-defined by setting a minimum mass ratio, similarly to what was done in these experiments. However, it also depends on the quality of the histograms provided. Table 7.4 denotes (5) that the multidimensional discretization of the explanatory variables has a considerable effect on the prediction's quality. This reduction of the target variable's variability is explained on the example provided in Fig. 7.7 (where it is possible to reduce the initial number of modes to just one).

Despite its contributions to the estimation of urban dynamics and related problems, the proposed methodology also presents two drawbacks: the aforementioned need to launch the `layer-off` from time to time, and the large amount of parameters. A sensitivity analysis was carried out on the most sensible subset of parameters, which strengthens its values. It is claimed that most parameters only have an impact on the granularity or reactiveness of the model. However, the truth is that its setting, even considering some *apriori* parameter fitting methodology, requires some previous experience on this problem.

It is also important to sustain that this framework does not address the presence of constraints (i.e. the river). This may cause clusters containing the two unconnected river margins to be formed. Fig. 7.6 exemplifies this undesirable effect, especially for ROI downtown. However, its effects are minimal in this specific study, which happens due to the high number of bridges in these regions (four), and due to their high density levels. To learn more about this topic, go to [Tung *et al.*, 2001].

This framework is applicable and/or adjustable to any urban analysis problem. Yet, it may not present a meaningful contribution to problems where the expected sample rate is large enough to employ batch learning models. However, this is not the case of real-time decision support systems, such as the recommendation models. Typically, their ability to produce accurate recommendations for the passenger finding problem depends on the production of **reliable** predictions on some dependent variables, such as the spatiotemporal distribution of the demand (as described on Chapter 6) and the regions' profitability [Yuan *et al.*, 2011b]. We want to claim this work as a **straightforward contribution** to maintain statistics of interest and/or induction models about the decision variables of real-time recommendation models on this topic, regardless of their target variable.

7.7 Related Work

The estimation of time-dependent O-D Matrices is a thrilling problem in many research areas. Each area may face the problem using different approaches, assumptions and ends. One of the most classical approaches is the real-time estimation of traffic flows on a freeway network. It consists of estimating the O-D flows using the real-time link counts and/or Automatic Vehicle Identification systems. The State-of-the-Art techniques to address this problem are Kalman Filters and the Generalized Least Squares [Dixon and Rilett, 2002; Zhou and Mahmassani, 2006; Zhu and others, 2007; Barceló *et al.*, 2013].

This work focuses on analyzing the O-D urban dynamics and it differs from the freeway-based approaches because it focuses on human behavioral patterns rather than on traffic modeling itself. Recently, a promising approach was also presented in [Zheng *et al.*, 2011]. This work also models urban mobility using O-D matrix matrices. However, they constrain the matrix boundaries to be major roads - which can be faced as a limitation of such approach.

The employment of ubiquitous (such as the ones provided by the taxi networks) rather than static sensors is also a key feature in this approach. The different discretization levels of this framework can be seen as an opportunity to maintain multiple statistics of interest of the O-D patterns instead of just one. The multidimensional tree-based discretization of the trips' attributes is straightforward on the real-time O-D matrix estimation problem, regardless of the research goal and scope.

7.8 Conclusions

This Chapter proposes a novel technique to maintain statistics regarding the relationships between Regions of Interest (ROI) in a urban area. This technique distinguishes itself from the existing State-of-the-Art because it employs **multiple discretization levels** over both the explanatory and target variables. Experiments conducted in a real-world case study validated its contributions in different aspects of this problem. Such experiments focused on a particular task (apriori Travel Time Estimation) using only one data source (i.e. taxi networks). However, this framework is prepared to handle other variables and multiple data sources. Its **incrementality** is the key of its adaptive characteristics. Due to these reasons, it represents a relevant contribution for those interested in inferring the future values of urban dynamic variables in **real-time**.

The three previous Chapters of this thesis addressed concrete Planning and Control problems on PT networks. However, this framework has a wider scope by addressing a more **fundamental** problem. Concretely, it focuses on a most general topic that is directly related with this thesis: How can we take full advantage on the multiple high-speed GPS data streams that are being produced on an urban environment? By answering this question, we intended to provide a sustainable way to handle these large amount of data in order to extract usable information from it. Such information is valuable in many research topics. It will be key to maintain the current quality levels of human mobility on the major metropolis worldwide.

As many other incremental frameworks, the error introduced by the continuous approximations performed by the different discretization levels make it necessary to maintain an **offline** operator which may be triggered from time to time to reduce the error. The most relevant aspect of the error introduced is the absence of an **online density refinement** of the mass-based clusters obtained through split/merge operations. Density-based spatial clustering algorithms are seen as promising approaches to address this issue. However, it is not possible to confirm if they are directly applicable to this specific context. This problem is an open research question.

Part IV

Concluding Remarks

Chapter 8

Conclusions

Today, there are multiple sources of rich spatiotemporal data related with the human mobility in the major urban areas. One of the most well known examples of such sources is the GPS. This location-based data contain patterns that can lead to a global optimization of the way that people can actual travel from one point to another. Such optimization can provoke multiple benefits from both passengers and mass transit agencies. In this thesis, we are focused on improving the profitability of such transit agencies by mining the GPS data broadcasted by their fleets (namely, taxis and buses networks). These vehicular networks provided an unprecedented opportunity of **learning** the human mobility behavior. Both comprise real-time sources of spatiotemporal data with a **high level of detail** whose, together, meet no parallel in the current literature. Consequently, our ultimate goal is to take advantage of the unique characteristics of these data sources to improve their operations. More than performing data analysis to uncover useful information to support the decision makers, we aimed to perform it in **real-time**.

To accomplish such goal, we undertook an explorative approach. We started it so with an analysis to the current State-of-the-Art to identify research opportunities to be explored by mining the GPS data broadcasted by these vehicles. The ultimate goal is to provide sustainable frameworks, from the computational point of view, to deal with such massive amounts of data in order to extract as much as possible usable information from them. As result, twelve specific research topics were drawn from such review (which are identified in Section 2.3). Six of these topics were covered by the research goals proposed to this thesis (defined in Section 2.4). The research performed to accomplish such goals provided contributions that resulted on a total of **sixteen publications** (two are still under peer review). They are ready to be deployed on any mass transit agency possessing a fleet equipped with GPS devices (on the case of the Control frameworks) and/or containing large-scale historical traces of their operations (to perform Planning tasks).

This Section starts by summarizing these contributions, to then describe how these goals were accomplished, as well as the publications that resulted from such contributions. Finally, some future research lines are pointed out.

8.1 Contributions

This thesis provided multiple contributions on this topic. These contributions were already described throughout this document. Even so, they are summarized as follows.

The Part I provides an overview of the problematic that we are addressing on this thesis divided on three chapters. It (1) starts with a brief introduction on the basics about public transportation problems and spatiotemporal data. Then, (2) a survey on the current State-of-the-Art on data driven methods for Planning and Control on Public Transportation Networks is provided in Chapter 2. It allowed to identify multiple issues on the current systems that could be improved by mining GPS data giving certainly opportunities of research. It concluded that it is mostly the Operational Control that is benefiting from this technology - while the Planning tasks are commonly carried out using traditional methods on transportation engineering. Hence, there are multiple research opportunities provided by this type of spatiotemporal data that were identified on this study. Some of them were already addressed in this study. Nevertheless, the Chapter 2 can be faced as a landmark review from which other researchers can build their work on. This Part ends with Chapter 3, where an overview of the online learning techniques and methods is provided.

The problematics about the Planning and Control on Mass Transit agencies studied on this thesis are detailed throughout the Part II - which contains two chapters. We departed from a (4) more persistent planning problem concerning the evaluation of the Schedule Plan's Coverage to then address (5) a real-time data driven Control framework able to tackle every kind of sporadic issues.

The SP coverage evaluation (4) concerns to assess whether the schedule coverage, in terms of the days covered by each schedule (e.g. Saturdays and Sundays, Workdays and Holidays), still meets the network behavior. Chapter 4 describes a ML framework which explores the variances of the round-trip times. It does it so by grouping each one of the days available into one of the possible coverage sets. This grouping is made according to a distance measured between each pair of days where the criteria rely on their profiles. As output, rules about which days should be covered by the same timetables are provided. Such rules can be used by the operational transportation planners to perform the above-mentioned evaluation. These rules also provide insights on how can the current coverage be changed in order to achieve that.

Chapter 4 presents an Automatic Control framework to mitigate the formation of (5) Bus Bunching in real-time. The framework depicts a powerful combination of State-of-the-art tools and methodologies such as Regression Analysis, Probabilistic Reasoning and Perceptron. The prediction's output is then used to select and deploy a corrective action (e.g. stop skipping) to automatically prevent bus bunching. Simulation results demonstrate that this method could eliminate most of the bunching occurrences and still decrease average passenger waiting times without prolonging in-vehicle times.

Both frameworks are ready to be deployed on any mass transit agency that have historical spatiotemporal data and fleets equipped with a full Automated Data Collection system. They can be used together or standalone and they meet no parallel on the literature by the novel types of information that they can provide about the network behavior.

The problem approached in Chapter 4 uncovered the need of monitoring closely the regularities in the human behavior in order to antecipate demand patterns such as recurrent (i.e. under certain conditions) peaks or valleys. This problematic is deeply approached in Part III, which analyses the urban mobility problems through a taxi company's perspective. Throughout the two Chapters of this Part, we study the taxi stand recommendation problem. It concerns not only the (6) short-term demand but also their profit and (7) the travel time prediction between different city areas.

Chapter 5 depicts a predictive framework to typify the short-term taxi-passenger demand over an urban area. It does so by (1) predicting the number of services to be demanded in each area/stand as well as their (2) fare-based profitability. This framework uses time series analysis methods such as non-homogeneous Poisson processes and ARIMA to perform such estimation on the demand's future values. It accomplished a low error rate when evaluated using real world data. Its adaptive characteristics represent an advance to the current State-of-the-Art because they allow to increase the flexibility of the service offer in order to successfully handle bursty demand peaks.

Chapter 6 introduces a novel three-step incremental framework to maintain statistics on the mobility dynamics over a time-evolving origin -destination (O-D) matrix. This framework intend to provide a sustainable way to handle these large amount of data in order to extract usable information from it independently of the problem we want to solve (namely, its variable of interest). It relies on multiple incremental methods which discretize the data over its multiple dimensions to approximate, as much as possible, the p.d.f. in place at each moment. The Travel Time Estimation (TTE) problem was regarded as a real-world application by predicting how much (vacant) time a Taxi Driver would take to go from a given city area to another after dropping-off a passenger. This work also enables the possibility of easily merge location-based data from multiple sources (e.g. buses, taxis, smartphones), independently of its spatial granularity.

All the abovementioned ML frameworks were successfully evaluated using data broadcasted by major bus and taxis operators running in the city of Porto, Portugal. The publications resulted from these contributions are described below.

8.2 Publications

This work resulted on a total of 16 publications: 5 journal papers (where two are still under peer review), 5 book chapters, 4 conference papers and 2 other workshop papers.

The Chapter 2 resulted into a survey in the IEEE Transactions on Intelligent Transportation Systems (IEEE TITS) [Moreira-Matias *et al.*, 2015]. The work of Chapter 4 was published on a conference paper published on the IEEE

International Conference on Intelligent Transportation Systems (ITSC) [Matias *et al.*, 2010] and extended later to a journal paper published on the Information Sciences Mendes-Moreira *et al.* [2015]. The framework described in Chapter 5 led to a total of 5 papers: firstly, an offline unsupervised learning technique was developed to analyze the causes of such BB events. Such work was published on a workshop Moreira-Matias *et al.* [2012c] and later, on a book chapter Moreira-Matias *et al.* [2012b]. Then, this work was extended to a real-time predictive methodology published on a book chapter Moreira-Matias *et al.* [2014a] and on a journal paper submitted to the Journal of Applied Soft Computing (and still under peer review). The results of this work were also published on the PhD spotlight session of European Conference of Machine Learning (ECML) together with some results of the Chapter 6 [Moreira-Matias *et al.*, 2014c]. Chapter six's work led to a total of 5 publications: an offline demand prediction method on the ITSC Moreira-Matias *et al.* [2012d] and a book chapter published by the International Symposium of Intelligent Data Analysis (IDA) Moreira-Matias *et al.* [2012e] where the ITSC's work is tested on a real-time environment. Then, this work was extended to a fully incremental methodology by suggesting the employment of first-order updates to compute the ARIMA's weights in the Portuguese Conference on Artificial Intelligence Moreira-Matias *et al.* [2013a] (book chapter). All this work was then summarized in a journal paper on IEEE TITS Moreira-Matias *et al.* [2013b]. Finally, The fare-based stand classification framework was recently published as a conference paper on ITSC Moreira-Matias *et al.* [2014b]. The Chapter 7 work led to one journal publication which is still under peer review on the Expert Systems with Applications. Two more publications were made to extend the work of Chapters 6 and 7 towards a real-time recommendation model: a conference paper on the IEEE Vehicular Network Conference (VNC) Moreira-Matias *et al.* [2012a] and also on a workshop paper on the International Joint Conference on Artificial Inteligence (IJCAI) Moreira-Matias *et al.* [2013c].

Throughout this thesis, some additional research was performed. Such works were also about applying ML frameworks to real world problems from different domains. Some examples are Wind Ramp Detection Ferreira *et al.* [2011], Text Mining Moreira-Matias *et al.* [2012f], Discretization methods for the Scheduling of Call Center's Agents Moreira-Matias *et al.* [2014d], on optimizing autonomous parking lots Nunes *et al.* [2014] or on data driven driver detection (a journal paper still under peer review on Transportation Research: Part F).

8.3 Goal Evaluation

Five primary goals were established to this thesis in Section 2.4. All were accomplished throughout this thesis. The real-time smart recommendations about the most adequate taxi stand to head to based on the current network status encloses the research goal which was not 100% accomplished. However, we must highlight that the ground breaking research is presented on this thesis while the work to be done is focused on secondary issues that, despite their relevance, do not comprise the major engine behind this problem.

As it was previously described in Section 2.2.2, such Recommendation relies on four variables from which three need to be predicted somehow. All these variables can be predicted using the frameworks introduced in Chapters 5 and 6. Any recommendation model will depend on the accuracy of these frameworks - which as been demonstrated to be high throughout this thesis on this specific context. Even so, we did not provided any evidence on how we could combine these multiple outputs on a single recommendation model (which is discussed in the next section).

The secondary goal was accomplished by the work described in Chapter 6. The time-evolving O-D matrix estimation framework can be fed by any source of spatiotemporal data concerning the human mobility, independently of the variable of interest and of the data granularity. Some examples on how this could be explored are provided as follows: (1) to estimate flow counts by joining up the AVL/APC data broadcasted by the buses to the taxi GPS traces or to (2) simply use the AVL data of buses traversing non-stop route sections (i.e. without bus stops) with the taxi traces to perform travel time estimation. Moreover, the results presented in Section 7.5 already demonstrated that this framework increases its accuracy along the amount of data available to it (i.e. by increasing the zoom level on the multidimensional attribute discretization). However, such concept still requires a proper proof of concept to demonstrate its complete validity.

8.4 Future Work

Today, we live in a true *Big Data* era. The problem is not how to obtain the data anymore...but how we **handle** the data that we have. As the value of the data availability goes down (ex.: many transportation companies already publish their AVL data for free access on the web [Dublinked, 2015; Beijing City Lab, 2015]), the value of the information we are able to extract from it rises at a same rate. It happens so because it is getting harder to mine all these sources of information on a sustainable way.

This thesis is focused on Public Transportation problems. However, some of the methods developed can be applied to a wider range of problems (e.g.: web traffic management using the work of Chapter 7). Moreover, we approach the problematic of mixing multiple data sources into one single framework (also known as *Data Fusion*). Such topic provide ground-breaking opportunities for the data mining research community throughout the next few years (e.g. on the estimation of time-evolving O-D matrices).

Throughout this document, promising opportunities and issues to be solved on future research were already identified (see, for instance, Section 2.3). It is possible to summarize them into five large areas: (1) parametrization, (2) evaluation, (3) framework development, (4) feature generation and (5) recommendation. Specific topics on each one of this areas are pointed out below, followed by some final remarks.

8.4.1 Parametrization

1. How can we determine the optimal number of schedules k to build a Schedule Plan given the current network and demand behavior (i.e. AVL data)?

2. How can we perform a dynamic setting of the parameters η (i.e. an headway-based minimum threshold to consider a BB event) and χ (i.e. a minimum BB likelihood threshold to deploy a corrective action) for each individual route and possibly also time-dependent (i.e. $\eta(t, \texttt{route})$, $\chi(t, \texttt{route})$)?

3. How is it possible to define a good granularity for each case study of interest on the fare-based model (i.e. histogram parameters, discretization rule set, etc.)?

8.4.2 Evaluation

1. To create an unique, integrated and global evaluation indicator on the SP reliability considering the company's perspective on the evaluation by including external factors in the evaluations, or by developing cost-related evaluations;

2. Evaluating the changes performed on the SP is difficult prior to deployment. Even there are various works focused on improving the SP, not many of them evaluate the impact of the suggested changes. The *before-and-after* evaluation studies are crucial to **quantify** the relevance of these adjustments.

3. To evaluate the taxi-passenger demand model on Scenario 2 urban areas, where the demand is larger than the service offer.

8.4.3 Framework Development

1. A large gap identified in the literature has to do with the AVL-based long-term TTP. The regression models represent the most relevant slice of the State-of-the-Art on AVL-based short-term TTP. However, some works have also demonstrated their usefulness in long-term problems [Mendes-Moreira *et al.*, 2012]. The AVL data makes it possible to explore these models to improve the SP (e.g. the timetabling or the driver's scheduling).

2. Slack time is a well-known technique to accommodate travel time variability. The availability and the reliability of the historical AVL data used today represent a clear opportunity to improve the schedules using this well-known strategy by, for instance, setting optimal slack times to each trip.

3. The work on Chapter 7 relies on isothetic boundaries for the spatial cluster definition. Is it possible to relax this constrain to consider more realistic approaches to the real ROI shapes and still maintain their incrementality?

4. Is it possible to also consider density-based approaches to improve the spatial clustering framework in place on the O-D matrix estimation framework?

8.4.4 Feature Generation

1. Feature Selection and Generation are important building blocks of any regression analysis. However, there is not much research on performing this task specifically for TTP. This is significantly important when employing some type of regression algorithms (such as SVR and ANN), which are highly sensitive on the feature set.

2. The Multidimensional Discretization method suggested in Chapter 7 used a predefined hierarchy of features. However, their relevance could also be mined from the data directly by using, for instance, Principal Component Analysis.

8.4.5 Recommendation

1. Most AVL-based works on improving Operational Planning (OP) on Mass Transit Agencies focus on the Schedule Plan. The AVL data makes it possible to perform a bottom-up OP evaluation, namely correctly exploring the available resources or even reducing them if possible to meet the current demand. A complete AVL-based framework to **re-design all the steps of the OP** is a research goal on this topic for the medium term future.

2. How can we combine all the decision variables on taxi stand choice problem to obtain an integrated recommendation model? One possibility is to assign a **score** to each stand k computed as linear combination of the values of such variables in the current instant (i.e. $RS_k(t) = \sum_i = Pr_i(t) \times w_i$: $\sum_i = w_i = 1$ where $Pr_i(t)$ denotes the variable's values on each moment and w_i denote their relevance, computed as an weight). Consequently, the problem would be to obtain such weights.

3. On these dynamic problems, the weight sets associated to the decision variables are commonly time dependent (i.e. $w_i(t)$). An initial approach to such problem was performed in [Moreira-Matias *et al.*, 2012a] by assigning $w_i(t)$ with a normalized version of each prediction residuals. However, this work just uses demand-based information to perform a proof of concept of its utility. Further experiments are required to demonstrate its validity.

8.5 Final Remarks

Novel challenges awaits the data enthusiasts. The speed of data communication and its rising quantity push the Machine Learning methods beyond unprecedented borders. Now, it is not enough to extract information from data in real-time. We need to be able to quantify its relevance at each moment for every decision making process - and, obviously, to continuously learn from their outcomes. Moreover, we have to do it combining multiple sources such as different vehicles, devices and types of information (e.g. weather, telecommunications and shopping records, etc.). This thesis provided surveys, data driven methods and mathematical formulations that minimized the distance between such futuristic vision and the present reality. Its impact on the Public Transportation problems approached in this document is undeniable. These solutions are the novel State-of-the-Art on such topics.

Appendices

Appendix A

Source Code of Consensual Clustering

```
consensusMatrix=function(cl)
{
 len=length(cl$cluster)
 m< matrix(0,len,len)
 for(i in 1:len)
 {
  for(j in 1:len)
  {
   if(cl$cluster[i]==cl$cluster[j])
   {
    m[i,j]=1
   }
  }
 }
 return (m)
}
list_matrix_dtw=function(X)
{
 m< matrix(0,365,356)
 ls_mdtw< list()

 for(j in X)
 {
  s< sprintf("%s.csv",j)
  orig.data< read.csv(s)
  new.data< Gera(orig.data)
  ls_mdtw< c(ls_mdtw, list(DTW(orig.data,new.data)))
  s< sprintf("%s_processed",j)
  print(s)

 }
 return (ls_mdtw)

}

```

```
generate_consensus_cluster=function(ls_mdtw,X,h,maxK,PATH)
{
 s < sprintf("%s.csv",X[1])
 orig.data < read.csv(s)
 cluster_lst < list()
 for(k in 2:maxK)
 {
  i < 0
  m< matrix(0,365,365)
  for(j in 1:length(ls_mdtw))
  {
   for(l in 1:h)
   {

    cl < kmeans(ls_mdtw[[j]],k,nstart=10)
    m<m+consensusMatrix(cl)
    i < i+1
   }
  }

  m<m/i
  m< 1 m


  dt < sprintf("")
  s < sprintf("%sConsensualDayDistribution.txt",PATH)

  if (k==2)
   write(dt,file=s,append=FALSE)

  cl < kmeans(m,k,nstart=10)
  cluster_lst < c(cluster_lst,list(cl))
  d< dayDistribution(cl,orig.data)
  write.table(d,file=s,col.names=TRUE,append=TRUE,sep="\t",row.names=TRUE)
 }

 dt < sprintf("Legenda: SEGUNDA_FE,TER,QUA,QUI,SEX,SAB,DOM > Numero de Dias da Semana no Cluster; FER,PON,NOR,TOL > Tipo de Dias no Cluster; FDS > Fins de Semana no Cluster; SP > Dias da Semana e de Fim de Semana em Periodo Nao Escolar; DTO > Total de Dias no Cluster")
 write(dt,file=s,append=TRUE)
 return(cluster_lst)
 #consensus< generate_consensus_cluster(ls_mdtw,c("300_1_1","301_1_1","205_1_1","205_1_2","505_1_1","505_1_2"),10,4,"T:\\")

}
consensus_cluster_bus=function(X,h,maxK,PATH)
{
 ls_mdtw< list_matrix_dtw(X)
 consensus < generate_consensus_cluster(ls_mdtw,X,h,maxK,PATH)
 return (consensus)
```

```
}



graph_bus=function(CLUSTER,PATH)
{
 #1264x695
 n<length(unique(CLUSTER$cluster))
 s<sprintf("%sfigurek=%03d.png",PATH,n)
 png(filename=s, height=695, width=1264,  bg="white")
 len<length(CLUSTER$cluster)
 mes<c(31,28,31,30,31,30,31,31,30,31,30,31)
 len_a<round(len/31)
 plot(1:length(CLUSTER$cluster),CLUSTER$cluster,type="l")
 a=1;for (i in 1:len_a) {points(a:(a+mes[i]),CLUSTER$cluster[a
     :(a+mes[i])],col=i);a<a+mes[i]}
 dev.off()
}



saveData=function(DATA,SOURCE,ARR,PATH,BUS_ID,ALGORITM)
{
 dt<sprintf("")
 s<sprintf("%s%sdayDistribution.txt",PATH,BUS_ID)
 write(dt,file=s,append=FALSE)
 if(ALGORITM=="consensus")
 {
  library(ConsensusClusterPlus)
  results<ConsensusClusterPlus(SOURCE,maxK=7,reps=50,pItem
      =0.8,pFeature=1)
 }
 for(i in ARR)
 {
  sk<sprintf("%s%skmeansk=%d.txt",PATH,BUS_ID,i)
  if(ALGORITM=="kmeans")
  {
   k<kmeans(SOURCE,i)
  }
  else
  {
   k<data.frame(cluster=results[[i]][["consensusClass"]])
  }
  d<dayDistribution(k,DATA)
  write.table(d,file=s,col.names=TRUE,append=TRUE,sep="\t",row
      .names=TRUE)
  sg<sprintf("%s%s",PATH,BUS_ID)
  graph_bus(k,sg)
  dt<data.frame(DAYS=c(1:length(k$cluster)),CLUSTER=k$cluster
      ,CENTERS=k$centers[k$cluster],SUM_SQERR=k$withinss[k$
      cluster])
  write.table(dt,file=sk,col.names=TRUE,append=FALSE,sep="\t",
      row.names=FALSE)
 }
```

```
 #return (data.frame(SEGUNDA_FE=segunda_feira,TER=terca_feira, QUA=quarta_feira, QUI=quinta_feira, SEX=sexta_feira, SAB=sabado,DOM=domingo,FER=feriado,PON=ponte,NOR=normal,TOL=tolerancia,FDS=sabado+domingo,SP=dias_semana_periodo_ne, FP=dias_fds_periodo_ne,DTO=total))
 dt< sprintf("Legenda:␣SEGUNDA_FE,TER,QUA,QUI,SEX,SAB,DOM> Numero␣de␣Dias␣da␣Semana␣no␣Cluster;␣FER,PON,NOR,TOL> Tipo␣de␣Dias␣no␣Cluster;␣FDS>Fins␣de␣Semana␣no␣Cluster;␣SP>Dias␣da␣Semana␣e␣de␣Fim de Semana␣em␣Periodo␣Nao␣Escolar;␣DTO>␣Total␣de␣Dias␣no␣Cluster")
 write(dt,file=s,append=TRUE)
}

generateData=function(DATA,ARR,PATH,BUS_ID)
{
 new.data< Gera(DATA)
 matrix.dtw<DTW(DATA,new.data)
 saveData(DATA,matrix.dtw,ARR,PATH,BUS_ID,"kmeans")
 return (matrix.dtw)
}

contains=function(CLUSTER,DATA,CLUSTERNUMBER)
{
  lista< c()
  vec< which(CLUSTER$cluster==CLUSTERNUMBER)
  for (j in 1:length(vec))
  {
   day< c(unique(DATA$DiaSemana[DATA$DiaAno==vec[j]]))
    lista< c(lista,day)
   day< c(unique(DATA$TipoDia[DATA$DiaAno==vec[j]]))
   day< day+7
    lista< c(lista,day)

  }
  vec1< vec
  #Periodo nao lectivo
  vec< vec[vec>(31+28+31+30+31+30+14) & vec <(31+28+31+30+31+30+31+31+15)]
  for (j in 1:length(vec))
  {
   aux< c(unique(DATA$DiaSemana[DATA$DiaAno==vec[j]]))
    if (length(aux)>0 & !is.na(aux[1]))
   {
     if (aux[1]==4) n< 13
     if (aux[1]==1) n< 13
     if (aux[1]>1 & aux[1]<4) n< 12
     if (aux[1]>4 & aux[1]<8) n< 12
     lista< c(lista,c(n))
   }
  }
  #pascoa
  vec2< vec1[vec1>89 & vec1<98]
  for (j in 1:length(vec2))
  {
```

```
    aux<c(unique(DATA$DiaSemana[DATA$DiaAno==vec2[j]]))
    if (length(aux)>0 & !is.na(aux[1]))
    {
     lista<c(lista,c(15))
    }
  }
  #natal
  vec2<vec1[vec1>355 & vec1<366]
  for (j in 1:length(vec2))
  {
    aux<c(unique(DATA$DiaSemana[DATA$DiaAno==vec2[j]]))
    if (length(aux)>0 & !is.na(aux[1]))
    {
     lista<c(lista,c(16))
    }
  }
  return (lista)
}

dayDistribution=function(CLUSTER,DATA)
{
 segunda_feira<c()
 terca_feira<c()
 quarta_feira<c()
 quinta_feira<c()
 sexta_feira<c()
 sabado<c()
 domingo<c()
 feriado<c()
 normal<c()
 ponte<c()
 tolerancia<c()
 total<c()
 dias_semana_periodo_ne<c()
 dias_fds_periodo_ne<c()
 dias_natal<c()
 dias_pascoa<c()
 for(j in 1:length(unique(CLUSTER$cluster)))
 {
    lista<contains(CLUSTER,DATA,j)
    segunda_feira<c(segunda_feira,sum(lista==5))
    terca_feira<c(terca_feira,sum(lista==7))
    quarta_feira<c(quarta_feira,sum(lista==2))
    quinta_feira<c(quinta_feira,sum(lista==3))
    sexta_feira<c(sexta_feira,sum(lista==6))
    sabado<c(sabado,sum(lista==4))
    domingo<c(domingo,sum(lista==1))

    feriado<c(feriado,sum(lista==8))
    ponte<c(ponte,sum(lista==11))
    normal<c(normal,sum(lista==9))
    tolerancia<c(tolerancia,sum(lista==10))
    total<c(total,sum(lista<8))
```

```
    dias_semana_periodo_ne < c(dias_semana_periodo_ne,sum(lista==12))
    dias_fds_periodo_ne < c(dias_fds_periodo_ne,sum(lista==13))
    dias_natal < c(dias_natal,sum(lista==16))
    dias_pascoa < c(dias_pascoa,sum(lista==15))
  }
  return (data.frame(SEGUNDA_FE=segunda_feira,TER=terca_feira, QUA=quarta_feira, QUI=quinta_feira, SEX=sexta_feira, SAB=sabado,DOM=domingo,FER=feriado,PON=ponte,NOR=normal,TOL=tolerancia,FDS=sabado+domingo,SP=dias_semana_periodo_ne, FP=dias_fds_periodo_ne,DNA=dias_natal,DPA=dias_pascoa,DTO=total))
}

profile_type=function(ORIGDATA,NEWDATA, dia)
{
 k < 0
 tipoDia < c()
 dia < (dia%%7)+7
 while(length(tipoDia)==0)
 {
  dia < dia+k
  tipoDia < c(unique(ORIGDATA$DiaSemana[ORIGDATA$DiaAno==dia]))
  k < k+7
 }
 vecDiasMesmoTipo < c(unique(ORIGDATA$DiaAno[c(ORIGDATA$DiaSemana)==tipoDia[1]]))
 vecDiasMesmoTipo < vecDiasMesmoTipo[vecDiasMesmoTipo<366]
 n< c()
 for(j in vecDiasMesmoTipo)
 {
  serie <NEWDATA[j,][!is.na(NEWDATA[j,])]

  if (length(serie)>3)
  {
   n < c(n,length(serie))
  }
 }

 n < round(mean(n))
 novaserie < c(1:n)*0
 dimnserie < c(1:n)*0

 for(j in vecDiasMesmoTipo)
 {
  serie <NEWDATA[j,][!is.na(NEWDATA[j,])]
  if (length(serie)>3)
  {
   pos < 0
   for (i in serie)
   {
    novaserie[pos] < novaserie[pos]+serie[pos]
    dimnserie[pos] < dimnserie[pos]+1
    pos < pos+1
```

```
      }
    }
  }
  n< dimnserie[dimnserie>0]
  serie< c()
  for(i in 1:length(dimnserie))
  {
    serie< c(serie,round(novaserie[i]/dimnserie[i]))
  }
  return(serie[!is.na(serie)])
  #matrix.dtw< generateData(orig.data,c(2:7),"T:\\")
}

DTW=function(ORIGDATA,NEWDATA)
{
  library(dtw)
  new< matrix(0,nrow=length(NEWDATA[,1]),ncol=length(NEWDATA
        [,1]),byrow=TRUE)
  x< 1
  while(x<=length(NEWDATA[,1]))
  {
    y< 1
    serie1 <NEWDATA[x,][!is.na(NEWDATA[x,])]
    while(y<=length(NEWDATA[,1]))
    {
      serie2 <NEWDATA[y,][!is.na(NEWDATA[y,])]
      if (sum(!is.na(serie1))<3)
      {
        serie1 < profile_type(ORIGDATA,NEWDATA,x)

      }
      if (sum(!is.na(serie2))<3)
      {
        serie2 < profile_type(ORIGDATA,NEWDATA,y)

      }
      if (new[x,y]==0)
        new[x,y] < dtw(serie1,serie2)$distance
      print(c(x,y))
      y< y+1
    }
    x< x+1
  }
  #matrix normalization
  new< new/mean(new)
  return (new)
}

Gera=function(X)
{
  nr< min(max(X$DiaAno),365)
  new.list < matrix(nrow=nr,ncol=100,byrow=TRUE)
  diaActual< 0
  viagem< 0
```

```
  viagem.dia<0
  for(dia in X$DiaAno)
  {
   viagem<viagem+1
   viagem.dia<viagem.dia+1
   if(dia>diaActual)
   {
    if(diaActual>0)
    {
     viagem.dia<1
    }
    if(diaActual==nr)
    {
     return (new.list)
    }
    diaActual<dia
   }
   new.list[diaActual,viagem.dia]<X$Duracao[viagem]
  }
  remove(viagem)
  remove(viagem.dia)
  remove(diaActual)
  remove(dia)
  return(new.list)
  remove(new.list)
}


median_profile_chart<function(routes)
{
   for (rts in routes)
   {
    filename<sprintf("%s%s%s","median_profile_",rts,".eps")#height=842,width=1500,
    postscript(filename,bg="white",horizontal=FALSE,onefile = FALSE, paper = "special",height=9, width=13)
    #pdf(filename,bg="white",width=13,height=9)
    par(oma=c(1.5,2.5,1,0))
    filename<sprintf("%s%s",rts,".csv")
    data<read.csv(file=filename,sep=",")

    data$Data<as.character(data$Data)
    data$DiaSemana<as.character(data$DiaSemana)
    data$TipoDia<as.character(data$TipoDia)

    data<data[data$DiaSemana!="domingo␣␣␣␣␣␣" & data$DiaSemana!="sabado␣␣␣␣␣␣␣" & data$TipoDia=="normal",]
    dias<unique(data$Data)
    id<1

    #min_y<min(data$Duracao)
    min_y<2000
    max_y<max(data$Duracao)
    #max_y<4500
```

```

  #min_x < min(data$InicioViagem)
  min_x < 20000
  #max_x < max(data$InicioViagem)
  max_x < 81000

  y < rep("",24)
  x < y
  for (i in c(1:24))
  {
 y[i] < to_string(i,"h")
 x[i] < to_string(i*5,"m")
 }



  #linhas das viagens
  for (dt in dias)
  {
 my_day < data[data$Data==dt,]
 my_day < my_day[with(my_day, order(InicioViagem)),]
 my_x < my_day$InicioViagem
 my_y < my_day$Duracao
 #str(my_x)
 #str(my_y)
 if (id==1)
 {
   plot(x=my_x, y=my_y, type="l", col="yellow2", ylim=c(min_y,max
      _y), xlim=c(20000,max_x), main="PROFILE OF THE WORKDAYS
      (mean & median)", axes="false", cex.main=2, xlab="", ylab
      ="")
   axis(1,c(1:24)*3600,y,cex.axis=1.4)
   mtext("Travel Start Time (in hours)", side=1, col="black",
      line=3.5, cex=2)
   axis(2,c(1:24)*300,x,cex.axis=1.4)
   mtext("Round Trip Time (in minutes)", side=2, col="black",
      line=3.5, cex=2)
 }
 else
 {
   lines(x=my_x, y=my_y, type="l", col="yellow2")
 }
 id < id+1
  }

  #bins
  max < max_x
  min < min_x
  nbins < 25

  bins < seq(from=min, to=max, by=(max min)/nbins)
  step < ((max min)/nbins)/2
  time_day < bins[1:(nbins 1)]+step
  duration < time_day
```

```
    duration2 < time_day
    id < 1
    total < 0
    samples < 0
    for (i in c(1:(nbins 1)))
    {
   #print(bins[i])
   tempos < data$Duracao[which(data$InicioViagem>=bins[i] & data
       $InicioViagem<=bins[i+1])]
   #print(tempos)
   duration[id] < median(tempos)
   duration2[id] < mean(tempos)
   total < total+sum(tempos)
   samples < samples+length(tempos)
   id < id+1
    }
    lines(x=time_day,y=duration,type="l",col="black",lwd=2.0)
    lines(x=time_day,y=duration2,type="l",lty=2,col="blue",lwd
       =2.0)
    legend(list(x = 19000,y = 40*60),legend=c("Bus Trips in 
        multiple workdays","Median Profile of the trips","Mean 
        Profile of the trips"), cex=1.28, col=c("yellow2","
        black","blue"), bty="l",lty=c(1,1,2),lwd=c(1,2,2),horiz
        =TRUE)
    print(rts)
    print(total/samples)
    dev.off()
   }
}
```

Appendix B

Source Code of Bus Bunching Mitigation

```
#preprocessing for Random Forests
prepare.regression < function(D)
{
 LINK_STOPS< as.character(D$LINK_STOPS)
 LINK_STOPS[which(LINK_STOPS=="0")] < "0_0"
 LINK_STOPS< unlist(strsplit(LINK_STOPS,"[_]"))
 idx < which(names(D) %in% c("NR_ORDEM_PARAGEM","NR_PTPARAGEM",
      "ID_PARAGEM_STCP","DATE","HOUR","MINUTES","SECONDS","TRIP
      ","ALL_SECONDS","LINK_STOPS","NR_MAT"))
D<D[, idx]
 idx1 < seq(1,length(LINK_STOPS) 1,2)
 idx2 < seq(2,length(LINK_STOPS),2)
 link1 <LINK_STOPS[idx1]

 link2 <LINK_STOPS[idx2]

D< cbind(D,LINK_START=link1,LINK_END=link2)

D$LINK_START< as.factor(D$LINK_START)
D$LINK_END< as.factor(D$LINK_END)


 if (length(unique(D$LINK_START))>32)
 {
   stops < unique(as.character(D$LINK_START))
   stops1 < stops[1:30]
   stops2 < stops[31:length(stops)]

   LINK_START1< as.character(D$LINK_START)
   for (st in stops2)
     LINK_START1[which(as.character(LINK_START1)==st)] < "1"
   LINK_START2< as.character(D$LINK_START)
   for (st in stops1)
     LINK_START2[which(as.character(LINK_START2)==st)] < "1"

```

```
    stops < unique(as.character(D$LINK_END))
    stops1 < stops[1:30]
    stops2 < stops[31:length(stops)]

    LINK_END1 < as.character(D$LINK_END)
    for (st in stops2)
     LINK_END1[which(as.character(LINK_END1)==st)] < "1"
    LINK_END2 < as.character(D$LINK_END)
    for (st in stops1)
     LINK_END2[which(as.character(LINK_END2)==st)] < "1"

    idx < which(names(D) %in% c("LINK_START","LINK_END"))
    D<D[, idx]
    D< cbind(D, LINK_START=LINK_START1, LINK_START2=LINK_START2,
        LINK_END=LINK_END1, LINK_END2=LINK_END2)
    D$LINK_START1 < as.factor(D$LINK_START)
    D$LINK_END1 < as.factor(D$LINK_END)
    D$LINK_START2 < as.factor(D$LINK_START2)
    D$LINK_END2 < as.factor(D$LINK_END2)
  }

  D$NR_TURNO < as.factor(D$NR_TURNO)
  print(length(unique(D$NR_TURNO)))
  D$NR_VIAGEM < as.numeric(D$NR_VIAGEM)
  print(length(unique(D$NR_VIAGEM)))
  D$HOLIDAY < as.factor(D$HOLIDAY)
  print(length(unique(D$HOLIDAY)))
  D$DAYNUMBER < as.numeric(D$DAYNUMBER)
  D$WEEKDAY < as.factor(D$WEEKDAY)
  str(D)

  return(D)

}

#offline regression using Random Forests
link_time_offline_regression < function(line, way, npastdays=7,
    lib=NULL)
{
  filename < sprintf("Line%d_S%d.csv", line, way)
  D< read.csv2(filename)
  filename < sprintf("Line%d_S%d_prediction_traindays%d.csv",
      line, way, npastdays)
  dataset < prepare.regression(D)
  D< cbind(D, prediction_RF=rep(0, length(D[,2])), prediction_PPR=
      rep(0, length(D[,2])))

  libraries2(lib)

  testday < npastdays+1
  idx < which(dataset$DAYNUMBER<testday)
  D$prediction_RF[idx] < 0/0
  D$prediction_PPR[idx] < 0/0

```

```
while(testday<=max(dataset$DAYNUMBER))
{
 print("")
 print("")
 print(testday)
 print("")
 print("")

 entrou < 0
 while(entrou==0 || length(test[1,])==0)
 {
  entrou < 1
  train < dataset[which(dataset$DAYNUMBER<testday & dataset$DAYNUMBER>=(testday  npastdays)),]
  idx < which(as.character(train$LINK_START)!="0")
  train < train[idx,]
  test < dataset[which(dataset$DAYNUMBER==testday),]
  if (length(test[1,])==0)
  {
   print(sprintf("day %d non existent...",testday))
   testday < testday+1
  }

 }
 print("train set...")
 print(print(train[1:100,]))
 print(length(train[,1]))

 print("test set...")
 print(print(test[1:100,]))
 print(length(test[,1]))


 print("training RF...")
 model< randomForest(LINK_TIME ~ ., data=train, mtry=3,
    ntrees=10)

 print("testing RF...")
 res < predict(model, test)
 res[which(res<0)] < 0
 idx < which(D$DAYNUMBER==testday)
D$prediction_RF[idx] < res
 print(res)

 print("training PPR...")
 model< ppr(LINK_TIME ~ ., data=train, nterms=2, max.terms=5)
 print("testing PPR...")
 res < predict(model, test)
 res[which(res<0)] < 0
D$prediction_PPR[idx] < res
 print(res)
 if((testday%%5)==0)
 {
  write.csv2(D, filename)
```

```
    }
    testday < testday+1

  }

  idx < which(as.character(D$LINK_START)=="0")
  if(length(idx)>0)
   D<D[ idx ,]

  write.csv2(D, filename)
}

#compute bus bunching trips
generate_bunching < function(line ,way, freq_BB_ratio=0.25)
{
  filename < sprintf("Line%d_S%d.csv",line ,way)
 D< read.csv2(filename)
 D$ID_PARAGEM_STCP < as.character(D$ID_PARAGEM_STCP)
 D$DATE< as.character(D$DATE)
 D$WEEKDAY< as.character(D$WEEKDAY)
 D$LINK_STOPS < as.character(D$LINK_STOPS)
  bunching <D[which(D$LINK_STOP=="0") ,]

  idx < which(names(bunching) %in% c("NR_ORDEM_PARAGEM","NR_
      PTPARAGEM","ID_PARAGEM_STCP","LINK_STOPS","NR_MAT","LINK_
      TIME"))
  bunching < bunching[ , idx]
  bunching < cbind(bunching ,FREQUENCY=bunching$ALL_SECONDS,
      BUNCHING=rep("NO_BUNCHING",length(bunching[ ,1])),STOP=rep
      ("N/A",length(bunching[ ,1])))
  idx < which(names(bunching) %in% c("ALL_SECONDS"))
  bunching < bunching[ , idx]

  bunching$BUNCHING< as.character(bunching$BUNCHING)
  bunching$BUNCHING[1] < "N/A"
  bunching$STOP< as.character(bunching$STOP)
  bunching$FREQUENCY[2:length(bunching$FREQUENCY)] < bunching$
      FREQUENCY[2:length(bunching$FREQUENCY)]  bunching$
      FREQUENCY[1:(length(bunching$FREQUENCY) 1)]



  trips < unique(bunching$TRIP)
  for (idx in c(2:length(trips)))
  {
    trip1 < trips[idx 1]
    trip2 < trips[idx]
    print("")
    bunching_th <max(freq_BB_ratio*bunching$FREQUENCY[which(
        bunching$TRIP==trip2)], freq_BB_ratio*120)
    print(sprintf("Threshold: %.4f seconds",bunching_th))

    trip1 <D[which(D$TRIP==trip1) ,]
    trip2 <D[which(D$TRIP==trip2) ,]
```

```
trocou < 0
if (length(trip1[,1])<length(trip2[,1]))
{
 trip_aux < trip1
 trip1 < trip2
 trip2 < trip_aux
 trocou < 1
}


common_stops < unique(c(trip1$ID_PARAGEM_STCP, trip2$ID_
    PARAGEM_STCP))

disjoint_stops < unique(c(common_stops[which(!(common_stops %
    in% trip1$ID_PARAGEM_STCP))],common_stops[which(!(common
    _stops %in% trip2$ID_PARAGEM_STCP))]))

if (length(disjoint_stops)>0)
 common_stops < common_stops[which(!(common_stops %in%
    disjoint_stops))]



if (trocou==1)
{
 trip_aux < trip1
 trip1 < trip2
 trip2 < trip_aux
}

idx1 < which(trip1$ID_PARAGEM_STCP %in% common_stops)
stops1 < trip1$ID_PARAGEM_STCP[idx1]
idx2 < which(trip2$ID_PARAGEM_STCP %in% common_stops)
stops2 < trip2$ID_PARAGEM_STCP[idx2]


max_len < min(length(idx1),length(idx2))
idx1 < idx1[1:max_len]
idx2 < idx2[1:max_len]


headways < abs(trip2$ALL_SECONDS[idx2]  trip1$ALL_SECONDS[idx1
    ])
print("Common_stops:")
print(common_stops)
print("Headways:")
print(headways)
bunching_stops < which(headways<bunching_th)
if(length(bunching_stops)>0)
{
 idx_stop < bunching_stops[1]
 idx_stop < idx1[idx_stop]
 bunching$STOP[idx] < trip1$ID_PARAGEM_STCP[idx_stop]
 bunching$BUNCHING[idx] < "BUNCHING"
```

```
    }
    print("Result:")
    print(bunching[idx,])
    #x< scan()
  }
  filename=sprintf("LINE%d_S%d_bunching.csv",line,way)
  write.csv2(bunching,filename)
}

#trip based update
add_inter_travel_prediction< function(line,way,train_horiz,lib
    =NULL,folder=NULL,beta=10,threshold=60*60*60,max_beta_
    value=2.0,fast_reduction_ratio=0.5,large_error_ratio=0.25,
    step.ratio=0.3)
{
  libraries2(lib)
  beta_value< beta/1000
  print(folder)
  if (length(folder)==0)
    filename< sprintf("Line%d_S%d_prediction_traindays%d_updated
        _beta10_FINAL.csv",line,way,train_horiz)
  else
    filename< sprintf("%s/Line%d_S%d_prediction_traindays%d_
        updated_beta10_FINAL.csv",folder,line,way,train_horiz)

  print(filename)
  D< read.csv2(filename)
  D<D[, 1]



  for (i in c(1:length(D[1,])))
   D[,i] < as.character(D[,i])

  idx.numeric < which(names(D) %in% c("NR_TURNO","NR_VIAGEM","NR
      _ORDEM_PARAGEM","NR_PT_PARAGEM","NR_MAT","HOUR","MINUTES"
      ,"SECONDS","TRIP","DAYNUMBER","ALL_SECONDS","LINK_TIME","
      prediction_RF","prediction_PPR","prediction_delta_rule_RF
      ","prediction_sliding_window"))
  for (i in idx.numeric)
   D[,i] < as.numeric(D[,i])
  #feed with NAs

  idx.rem < which(names(D) %in% c("prediction_delta_rule_RF","
      prediction_delta_rule2_RF","prediction_delta_rule3_RF"))
  D<D[, idx.rem]

  D< cbind(D,prediction_delta_rule_RF=D$prediction_RF,
      prediction_sliding_window=D$prediction_RF)
  D$prediction_delta_rule_RF < as.numeric(D$prediction_delta_
      rule_RF)
  D$prediction_sliding_window < as.numeric(D$prediction_sliding_
      window)
  #get the trips
```

```
j < which(names(D)=="prediction_RF")
D<D[which(!is.na(D[,j])),]
trips < unique(D$TRIP)


bunching <D[which(D$LINK_STOP=="0"),]

idx < which(names(bunching) %in% c("NR_ORDEM_PARAGEM","NR_PTPARAGEM","ID_PARAGEM_STCP","LINK_STOPS","NR_MAT","LINK_TIME"))
bunching < bunching[, idx]
bunching < cbind(bunching,FREQUENCY=bunching$ALL_SECONDS)
idx < which(names(bunching) %in% c("ALL_SECONDS"))
bunching < bunching[, idx]

bunching$FREQUENCY< as.numeric(as.character(bunching$FREQUENCY))


bunching$FREQUENCY[2:length(bunching$FREQUENCY)] < bunching$FREQUENCY[2:length(bunching$FREQUENCY)] bunching$FREQUENCY[1:(length(bunching$FREQUENCY) 1)]

idx < which(names(bunching) %in% c("TRIP","FREQUENCY"))
frequencies < bunching[,idx]

frequencies$TRIP < as.numeric(frequencies$TRIP)
str(frequencies)

na_idx < which(is.na(D[,j]))
idx < rev(na_idx)[1]+1

base.step < 0.01
current.step < base.step
neg.counter < 0
pos.counter < 0

neg.counter2 < 0
pos.counter2 < 0
beta_value2 < beta_value
for (tr in (1:(length(trips) 1)))
{

  trip1 < trips[tr]
  trip2 < trips[tr+1]
  seltrip2 < trip2
  print("")

  trip1 <D[which(D$TRIP==trip1),]
  trip2 <D[which(D$TRIP==trip2),]
```

```

print("real")
print(trip1$LINK_TIME)
print("predicted")
pre<round(trip1$prediction_RF)
print(pre)
print("")
frequency< trip2$ALL_SECONDS[1]  trip1$ALL_SECONDS[1]
error_th< 0.25*frequency/10/2

#RF preprocessing
trip1$prediction_RF[1] < 0
sel.max< which(trip1$prediction_RF>3600)
if (length(sel.max)>0)
 trip1$prediction_RF[sel.max] < 3600

print("admissable␣error")
print(error_th)
print("")

print("RF")
arr< trip1$prediction_RF  trip1$LINK_TIME
arr< arr[which(!is.na(arr))]
print(arr)
simple.voting<mean(arr)
print(sum(abs(arr)))

#delta rule preprocessing
trip1$prediction_sliding_window[1] < 0
sel.max< which(trip1$prediction_sliding_window>3600)
if (length(sel.max)>0)
 trip1$prediction_sliding_window[sel.max] < 3600

trip1$prediction_sliding_window[1] < 0
sel.max< which(trip1$prediction_sliding_window>3600)
if (length(sel.max)>0)
 trip1$prediction_sliding_window[sel.max] < 3600


print("delta_rule")
arr2< trip1$prediction_delta_rule_RF  trip1$LINK_TIME
arr2< arr2[which(!is.na(arr2))]
print(arr2)
print(sum(abs(arr2)))

print("sliding_window_rule")
arr3< trip1$prediction_sliding_window  trip1$LINK_TIME
arr3< arr3[which(!is.na(arr3))]
print(arr3)
print(sum(abs(arr3)))


trip2$prediction_delta_rule_RF[1] < 0
sel.max< which(trip2$prediction_delta_rule_RF>3600)
```

```
if (length(sel.max)>0)
 trip2$prediction_delta_rule_RF[sel.max] < 3600

trip2$prediction_sliding_window< trip2$prediction_delta_rule_RF

trip2$prediction_delta_rule_RF< trip2$prediction_sliding_window
trip1$prediction_delta_rule_RF< trip1$prediction_sliding_window
neg< which(arr< ( 5))
pos< which(arr>5)
perc.neg< length(neg)/length(arr)
perc.pos< length(pos)/length(arr)

print("")
print("percs_neg")
print(perc.neg)
if (frequency<(120*60))
{
 if(!is.na(perc.neg) && !is.na(perc.pos) && perc.neg>perc.pos && perc.neg>0.3)
 {
  print("entrou")
  neg.counter< neg.counter+1
  trip2$prediction_delta_rule_RF< trip2$prediction_delta_rule_RF+(abs(trip2$prediction_delta_rule_RF)*(beta_value))
  beta_value< beta_value+(neg.counter*1.3)*base.step
 }


 print("percs_pos")
 print(perc.pos)

 if(!is.na(perc.neg) && !is.na(perc.pos) && perc.pos>perc.neg && perc.pos>0.3)
 {
  print("entrou")
  pos.counter< pos.counter+1
  trip2$prediction_delta_rule_RF< trip2$prediction_delta_rule_RF (abs(trip2$prediction_delta_rule_RF)*(beta_value))
  beta_value< beta_value+(pos.counter*1.3)*base.step
 }
}

if (is.na(perc.neg) || is.na(perc.pos) || !((perc.pos>perc.neg && perc.pos>0.3) || (perc.neg>perc.pos && perc.neg>0.3)) || sum(abs(arr2))>sum(abs(arr)) || frequency>=(120*60))
{
 pos.counter< 0
 neg.counter< 0
```

```
    beta_value < base.step
  }

  if (!is.na(simple.voting) && abs(simple.voting)>=error_th)
  {
   if(simple.voting>0)
   {
    print("entrou")
    neg.counter2 < neg.counter2+1
    trip2$prediction_sliding_window < trip2$prediction_sliding_window (abs(trip2$prediction_sliding_window)*(beta_value2))
    beta_value2 < beta_value2+(neg.counter2*1.3)*base.step

   }
   else
   {

    print("entrou")
    pos.counter2 < pos.counter2+1
    trip2$prediction_sliding_window < trip2$prediction_sliding_window+(abs(trip2$prediction_sliding_window)*(beta_value2))
    beta_value2 < beta_value2+(pos.counter2*1.3)*base.step
   }
  }
  else
  {
   pos.counter2 < 0
   neg.counter2 < 0
   beta_value2 < base.step

  }

  D$prediction_RF[which(D$TRIP==seltrip2)[1]] < 0
  D$prediction_delta_rule_RF[which(D$TRIP==seltrip2)] < trip2$prediction_delta_rule_RF
  D$prediction_sliding_window[which(D$TRIP==seltrip2)] < trip2$prediction_sliding_window
  beta_value < min(beta_value,0.3)
  beta_value2 < min(beta_value2,0.3)
  print(beta_value)
  print(beta_value2)

  if (tr%%10==0)
  {
   if (length(folder)==0)
```

```
      filename < sprintf("Line%d_S%d_prediction_traindays%d_
          updated_beta%d.csv",line,way,train_horiz,beta)
    else
      filename < sprintf("%s/Line%d_S%d_prediction_traindays%d_
          updated_beta%d.csv",folder,line,way,train_horiz,beta)

    write.csv2(D,filename)
   }
  }
  if (length(folder)==0)
    filename < sprintf("Line%d_S%d_prediction_traindays%d_updated
        _beta%d_FINAL2.csv",line,way,train_horiz,beta)
  else
    filename < sprintf("%s/Line%d_S%d_prediction_traindays%d_
        updated_beta%d_FINAL2.csv",folder,line,way,train_horiz,
        beta)

  write.csv2(D,filename)

}


#predict bunching and deploy control actions
predict_bunching < function(line,way,time="short",DO_ACTIONS="
    ACTIONS",first_trip_index=1,freq_BB_ratio=0.25,nstops=4,
    sliding_window_MAE_size=5,beta=0.1, max_frequency_pass=60,
    bus_capacity=150,new_stops_demand=4,perc_exits=0.2,max_
    passengers_stop=50,demand_var=0.2,PLOT_DEMAND=TRUE,prob_th
    _min=0.4,minimum_holding_time=30)
{

  #reading data and preprocessing
  filename < sprintf("Line%d_S%d_prediction_traindays7_updated_
      beta10_FINAL2.csv",line,way)
 D< read.csv2(filename)
 D$ID_PARAGEM_STCP < as.character(D$ID_PARAGEM_STCP)
 D$DATE< as.character(D$DATE)
 D$WEEKDAY< as.character(D$WEEKDAY)
 D$LINK_STOPS< as.character(D$LINK_STOPS)
  bunching <D[which(D$LINK_STOP=="0"),]

  idx < which(names(bunching) %in% c("NR_ORDEM_PARAGEM","NR_
      PTPARAGEM","ID_PARAGEM_STCP","LINK_STOPS","NR_MAT","
      prediction_RF","prediction_PPR","prediction_delta_rule_RF
      "))
  bunching < bunching[, idx]
  bunching < cbind(bunching,FREQUENCY=bunching$ALL_SECONDS,
      BUNCHING=rep("NO_BUNCHING",length(bunching[,1])),STOP=rep
      ("N/A",length(bunching[,1])),PROBABILITY1=rep(0,length(
      bunching$ALL_SECONDS)),PROBABILITY2=rep(0,length(bunching
      $ALL_SECONDS)),headway_MAE_delta_rule_online=rep(0,length
      (bunching$ALL_SECONDS)),headway_MAE_inter_trip=rep(0,
      length(bunching$ALL_SECONDS)),online_delta_rule=rep(0,
      length(bunching$ALL_SECONDS)),nstops=rep(0,length(
```

```
bunching$ALL_SECONDS)),stop_predicted=rep( 1,length(
    bunching$ALL_SECONDS)),stop_action=rep( 1,length(bunching
    $ALL_SECONDS)),stop_ocurred=rep( 1,length(bunching$ALL_
    SECONDS)),prediction_stop=rep( 1,length(bunching$ALL_
    SECONDS)),BUNCHING_ONLINE=rep("NO_BUNCHING",length(
    bunching[,1])),TWT=0,TIVT=0,TB1=0,TB2=0,ACTION=rep("N/A",
    length(bunching[,1])),time_amount=rep(0,length(bunching
    [,1])),RESULTED=rep(0,length(bunching[,1])))
idx < which(names(bunching) %in% c("ALL_SECONDS"))
bunching < bunching[, idx]

bunching$BUNCHING < as.character(bunching$BUNCHING)
bunching$BUNCHING_ONLINE < as.character(bunching$BUNCHING_
    ONLINE)
bunching$ACTION < as.character(bunching$ACTION)
bunching$BUNCHING[1] < "N/A"
bunching$ACTION[1] < "N/A"
bunching$BUNCHING_ONLINE[1] < "N/A"
bunching$STOP < as.character(bunching$STOP)
bunching$FREQUENCY[2:length(bunching$FREQUENCY)] < bunching$
    FREQUENCY[2:length(bunching$FREQUENCY)]  bunching$
    FREQUENCY[1:(length(bunching$FREQUENCY) 1)]

#variable initialization
TIVT < 0
TWT < 0
TB1 < 0
TB2 < 0
sample_pass_capacity1 < 10000000
ivts < rep(0,sample_pass_capacity1)
npass_stat1 < 0
sample_pass_capacity2 < 10000000
wts < rep(0,sample_pass_capacity2)
sample_pass_capacity3 < 10000000
headways_array < rep(0,sample_pass_capacity3)
plot_periodicity < 0
plot_periodicity2 < 0
npass_stat2 < 0
npass_stat3 < 0
chart_sample_periodicity < 5000

min_dwell_time < 10
time_for_boarding_per_passenger < 3

#syntentic demand matrix initialization
m< matrix("0",100000,4)
m< as.data.frame(m)
names(m) < c("STOP","ACTIV","LAST_UPDATED","PASSENGERS_WAITING
    ")
for (i in c(2:4))
 m[,i] < as.numeric(m[,i])
m[,1] < as.character(m[,1])
passengers_stops2 <m
n_pass_stops < 0
```

```
trips < unique(bunching$TRIP)
recent_maes < rep(20, sliding_window_MAE_size)

erro_paragem < rep(60,60)
n_erros_paragem < rep(1,60)
select_action < 0


idx < first_trip_index
trip_action < 0
cons < 0
if (time=="short")
 time < round(0.05*length(trips))
else
 time < length(trips)

start_time < as.numeric(proc.time()[3])

#for all trips
while (idx<time)
{
 #ghost trip
 if (select_action>0)
 {
  print("Action")
  print(select_action)
  print("Undoing...")
  #repeating last trip
  idx < idx 1
  npass_stat2 < old_npass_stat2
  npass_stat1 < old_npass_stat1
  passengers_stops2 < old.passengers_stops2

  cons < cons+1

 }

 old.passengers_stops2 < passengers_stops2
 idx < idx+1

 old_npass_stat2 < npass_stat2
 old_npass_stat1 < npass_stat1

 #getting pair of trips of interest
 trip1 < trips[idx 1]
 trip2 < trips[idx]
 print("")
 my_freq <max(bunching$FREQUENCY[which(bunching$TRIP==trip2)
     ],120)
 bunching_th <max(freq_BB_ratio*my_freq, freq_BB_ratio*120)
 bunching_th_sup < (freq_BB_ratio+0.05)*bunching$FREQUENCY[
     which(bunching$TRIP==trip2)]
```

```
  bunching_th_inf < (freq_BB_ratio 0.05)*bunching$FREQUENCY[which(bunching$TRIP==trip2)]
  print(sprintf("Threshold: %.4f seconds",bunching_th))

  trip1 <D[which(D$TRIP==trip1),]
  trip2 <D[which(D$TRIP==trip2),]
  trocou < 0
  if (length(trip1[,1])<length(trip2[,1]))
  {
   trip_aux < trip1
   trip1 < trip2
   trip2 < trip_aux
   trocou < 1
  }

  #avoiding missing data
  common_stops < unique(c(trip1$ID_PARAGEM_STCP, trip2$ID_PARAGEM_STCP))

  disjoint_stops < unique(c(common_stops[which(!(common_stops %in% trip1$ID_PARAGEM_STCP))],common_stops[which(!(common_stops %in% trip2$ID_PARAGEM_STCP))]))

  if (length(disjoint_stops)>0)
   common_stops < common_stops[which(!(common_stops %in% disjoint_stops))]


  if (trocou==1)
  {
   trip_aux < trip1
   trip1 < trip2
   trip2 < trip_aux
  }

  idx1 < which(trip1$ID_PARAGEM_STCP %in% common_stops)
  stops1 < trip1$ID_PARAGEM_STCP[idx1]
  idx2 < which(trip2$ID_PARAGEM_STCP %in% common_stops)
  stops2 < trip2$ID_PARAGEM_STCP[idx2]



  max_len < min(length(idx1),length(idx2))
  idx1 < idx1[1:max_len]
  idx2 < idx2[1:max_len]


  #calculate arrival times
  if(length(idx1)>1)
  {
   start1 < trip1$ALL_SECONDS[idx1][1]

   start_trip_day_seconds < trip1$HOUR[idx1][1]*3600+trip1$MINUTES[idx1][1]*60+trip1$SECONDS[idx1][1]
```

```
    start_day_date < trip1$DATE[1]
    start_weekday < trip1$WEEKDAY[1]
    year_date < substr(start_day_date,1,4)
    month_date < substr(start_day_date,6,7)
    day_date < substr(start_day_date,9,10)

    factores_procura < rep(2,length(common_stops)) (c(0:length(common_stops))*2/rep(length(common_stops),length(common_stops)))


    #parameters of the demand model
    expected_lambda < 90
    perc_freq_demand < 0.15
    expected_perc_route_completed < 0.25

    #bursty demand events (deactivated)
    bursty_stops < c()


    max_frequency_pass < min(max(my_freq*perc_freq_demand, expected_lambda),2*expected_lambda/3+expected_lambda)

    #realistic demand simulation with gaussian noise
    weights_demand < rnorm(length(factores_procura),max_frequency_pass,max_frequency_pass*demand_var)
    sel.correction < which(weights_demand<(expected_lambda/2))
    if(length(sel.correction)>0)
     weights_demand[sel.correction] < expected_lambda/2


    if(length(bursty_stops)>0)
     weights_demand[bursty_stops] < weights_demand[bursty_stops]*bursty_demand_quant



    bus_switch < 0

    for (i in c(1:length(common_stops)))
    {
     #demand generaton
     new_stops < common_stops[i]
     sel.stops < which(passengers_stops2$STOP==new_stops & passengers_stops2$ACTIV==1)
     if (length(sel.stops)>0 && !is.na(real_times_df$real_times[i]))
     {
      delta < max(0,real_times_df$real_times[i] passengers_stops2$LAST_UPDATED[sel.stops])

      if(is.na(delta) || is.na(max_frequency_pass))
      {
       #error
```

```
        print("NA1")
      }
      if(delta<(max_frequency_pass*60))
      {
        passengers_stops2$PASSENGERS_WAITING[sel.stops] < min(
            passengers_stops2$PASSENGERS_WAITING[sel.stops]+
            round(delta%/%weights_demand[i]*factores_procura[i])
            ,min(round(max_passengers_stop*factores_procura[idx
            ]),max_passengers_stop))
        if (i==length(weights_demand))
          passengers_stops2$PASSENGERS_WAITING[sel.stops] < 0
      }

      else
        passengers_stops2$PASSENGERS_WAITING[sel.stops] < min(
            round(new_stops_demand*factores_procura[i]),min(
            round(max_passengers_stop*factores_procura[idx]),max
            _passengers_stop))

      if (round(delta%/%weights_demand[i]*(factores_procura[i]/
          2))>30 || round(delta%/%weights_demand[i]*(factores_
          procura[i]/2))<0)
      {
        #error
        print("ALARME1")
      }

      passengers_stops2$LAST_UPDATED[sel.stops] < start1
    }
    else
    {
      n_pass_stops < n_pass_stops+1
      passengers_stops2$LAST_UPDATED[n_pass_stops] < start1
      passengers_stops2$ACTIV[n_pass_stops] < 1
      passengers_stops2$STOP[n_pass_stops] < new_stops
      passengers_stops2$PASSENGERS_WAITING[n_pass_stops]
      passengers_stops2$PASSENGERS_WAITING[n_pass_stops] < min(
          round(new_stops_demand*factores_procura[i]),max_
          passengers_stop)
    }
  }


  pre1 < trip1$prediction_sliding_window[idx1]
  all_seconds < rep(0,length(pre1))
  pre1[1] < start1
  for(i in c(2:length(pre1)))
    pre1[i] < pre1[i]+pre1[i 1]

  start2 < trip2$ALL_SECONDS[idx2][1]

  pre2 < trip2$prediction_sliding_window[idx2]
  all_seconds < rep(0,length(pre2))
  pre2[1] < start2
```

```
  for(i in c(2:length(pre2)))
   pre2[i]<pre2[i]+pre2[i 1]

  real_times1<trip1$LINK_TIME[idx1]
  real_times2<trip2$LINK_TIME[idx2]

  real_times1[1]<start1
  for(i in c(2:length(real_times1)))
   real_times1[i]<real_times1[i]+real_times1[i 1]

  real_times2[1]<start2
  for(i in c(2:length(real_times2)))
   real_times2[i]<real_times2[i]+real_times2[i 1]


  if(length(common_stops)!=length(real_times1))
  {
   real_times1<real_times1[1:length(common_stops)]
   real_times2<real_times2[1:length(common_stops)]
  }

  #deploying corrective actions on ghost trips
  if (trip_action==2 && select_action>0)
   real_times2[stops_action]<real_times2[stops_action]+amount_spread


  if (trip_action==1 && select_action>0)
   real_times1[stops_action]<real_times1[stops_action]+amount_spread


  arr<c(real_times1,real_times2)
  sel.nas<which(is.na(arr))
  if(length(sel.nas)>0)
   arr<arr[ sel.nas]

  #bunching detection
  real_times_df<data.frame(real_times=arr,bus=c(rep(1,length(arr)/2),rep(2,length(arr)/2)), stop=c(common_stops,common_stops))

  real_times_df$stop=as.character(common_stops)
  real_times_df<real_times_df[with(real_times_df, order(real_times,bus,stop)), ]


  bus1<rep(0,length(common_stops))
  bus2<rep(0,length(common_stops))

  pass_in_stops1<rep(0,length(common_stops))
  exits_in_stops1<rep(0,length(common_stops))
  pass_in_stops2<rep(0,length(common_stops))
  exits_in_stops2<rep(0,length(common_stops))
```

```
current_time< start_trip_day_seconds
current_date< start_day_date
idx_bus1< 0
idx_bus2< 0
idx
for (i in c(1:length(real_times_df[,1])))
{


 if(real_times_df$bus[i]==1)
 {
  my_bus< bus1
  idx_bus1< idx_bus1+1
  idx_bus< idx_bus1
  pass_in_stops< pass_in_stops1
  exits_in_stops< exits_in_stops1

 }
 else
 {
  if (bus_switch==0)
  {
   bus_switch< 1
   #backup to ghost trip
   passengers_stops< passengers_stops2
  }
  my_bus< bus2
  idx_bus2< idx_bus2+1
  idx_bus< idx_bus2
  pass_in_stops< pass_in_stops2
  exits_in_stops< exits_in_stops2

 }

 new_stops< real_times_df$stop[i]
 ns<new_stops
 sel.stops<which(passengers_stops2$STOP==new_stops &
     passengers_stops2$ACTIV==1)

 delta<max(real_times_df$real_times[i] passengers_stops2$
     LAST_UPDATED[sel.stops],0)

 if(is.na(delta) || is.na(max_frequency_pass))
 {
  print("NA2")


 }
 if(delta<(max_frequency_pass*60))
 {
  passengers_stops2$PASSENGERS_WAITING[sel.stops]<min(
      passengers_stops2$PASSENGERS_WAITING[sel.stops]+round
      (delta%/%weights_demand[idx_bus]*(factores_procura[
      idx_bus]/2)),min(round(max_passengers_stop*factores_
```

```
procura[idx_bus]),max_passengers_stop))
  if (i==length(weights_demand))
   passengers_stops2$PASSENGERS_WAITING[sel.stops] < 0
 }
 else
  passengers_stops2$PASSENGERS_WAITING[sel.stops] < min(round(new_stops_demand*factores_procura[idx_bus]),min(round(max_passengers_stop*factores_procura[idx_bus]),max_passengers_stop))

 if (round(delta%/%weights_demand[idx_bus]*(factores_procura[idx_bus]/2))>30 || round(delta%/%weights_demand[idx_bus]*(factores_procura[idx_bus]/2))<0)
 {
  print("ALARME2")

 }

 passengers_stops2$LAST_UPDATED[sel.stops] < real_times_df$real_times[i]



 #occupancy
 pass_in_stops[idx_bus] < passengers_stops2$PASSENGERS_WAITING[sel.stops]

 #outbounded demand corrections
 if (is.na(pass_in_stops[idx_bus]))
 {
  passengers_stops2$PASSENGERS_WAITING[sel.stops] < min(round(new_stops_demand*factores_procura[idx_bus]),min(round(max_passengers_stop*factores_procura[idx_bus]),max_passengers_stop))
  pass_in_stops[idx_bus] < passengers_stops2$PASSENGERS_WAITING[sel.stops]
 }
 if (idx_bus==length(factores_procura))
  pass_in_stops[idx_bus] < 0



 #deploying actions effects on consecutive trips
 if (real_times_df$bus[i]==2 && select_action>1 && (idx_bus %in% stops_action))
 {
  propagate_pass < (pass_in_stops[idx_bus] prev.pass_in_stops2[idx_bus])*3
  if ((propagate_pass>0 && select_action>=2))
  {
   print("old_times:")
   print(real_times2)
   real_times2[(idx_bus+1):length(real_times2)] < rep(propagate_pass,length(c((idx_bus+1):length(real_
```

```
        times2))))+real_times2[(idx_bus+1):length(real_
        times2)]


      sel.nas<which(is.na(real_times2))
      if(length(sel.nas)>0)
       real_times2<real_times2[ sel.nas]
      else
      {
       for (zz in c((i+1):length(real_times_df[,1])))
       {
        old.value<real_times_df$bus[zz]
        if(!is.na(real_times_df$bus[zz]) && real_times_df$bus[
            zz]==real_times_df$bus[i])
         real_times_df$real_times[zz]<real_times_df$real_
             times[zz]+propagate_pass
        if(is.na(real_times_df$bus[zz]))
         real_times_df$bus[zz]<old.value
       }
      }
      print("new_times:")
      print(real_times2)
     }
    }

    #deploying actions effects on consecutive trips
    if (real_times_df$bus[i]==1 && select_action>1 && (idx_bus
        %in% stops_action))
    {
     propagate_pass<(pass_in_stops[idx_bus] prev.pass_in_
         stops1[idx_bus])*time_for_boarding_per_passenger
     if ((propagate_pass<0 && select_action==3))
     {
      print("old_times:")
      print(real_times1)
      real_times1[(idx_bus+1):length(real_times1)]<rep(
          propagate_pass,length(c((idx_bus+1):length(real_
          times1))))+real_times1[(idx_bus+1):length(real_
          times1)]
      sel.nas<which(is.na(real_times1))
      if(length(sel.nas)>0)

       real_times1<real_times1[ sel.nas]
      else
      {
       for (zz in c((i+1):length(real_times_df[,1])))
       {
        old.value<real_times_df$bus[zz]
        if(!is.na(real_times_df$bus[zz]) && real_times_df$bus[
            zz]==real_times_df$bus[i])
         real_times_df$real_times[zz]<real_times_df$real_
             times[zz]+propagate_pass
        if(is.na(real_times_df$bus[zz]))
         real_times_df$bus[zz]<old.value
```

```
        }
      }
      print("new_times:")
      print(real_times1)
    }
  }

  #alightings generation...
  if (idx_bus>1)
  {
   if (idx_bus==length(exits_in_stops))
   {
    exits_in_stops[idx_bus]<my_bus[idx_bus 1]
   }
   else
   {
    stops_to_exit<round(rlnorm(pass_in_stops[idx_bus 1],log(max(expected_perc_route_completed*length(factores_procura)*factores_procura[idx_bus]*0.75,1)),log(max(1,expected_perc_route_completed*length(factores_procura)*0.2))))

    sel.stops.exit<which(stops_to_exit<1)
    if (length(sel.stops.exit)>0)
     stops_to_exit[sel.stops.exit]<1

    stops_to_exit< stops_to_exit+idx_bus 1
    sel.stops.exit<which(stops_to_exit>length(factores_procura))
    if (length(sel.stops.exit)>0)
     stops_to_exit[sel.stops.exit]<length(factores_procura)

    indexes<as.numeric(as.character(as.data.frame(table(stops_to_exit))$stops_to_exit))
    freqs<c(as.data.frame(table(stops_to_exit))$Freq)
    exits_in_stops[indexes]< exits_in_stops[indexes]+freqs

    if(select_action==3 && stops_action[1]==idx_bus && real_times_df$bus[i]==1)
    {
     exits_in_stops[idx_bus+1]< exits_in_stops[idx_bus+1]+exits_in_stops[idx_bus]
     exits_in_stops[idx_bus]<0
    }
    sel.nas<which(is.na(exits_in_stops))
    if (length(sel.nas)>0)
     exits_in_stops[sel.nas]<rev(freqs)[1]

   my_starting_stop<idx_bus 1

    for (jl in c(1:length(indexes)))
    {
     my_ending_stop<indexes[jl]

```

```
        if (pass_in_stops[idx_bus 1]>0 && length(my_ending_stop)==0)
         my_ending_stop < length(real_times1)
        else
        {
         if(pass_in_stops[idx_bus 1]>0)
         {
          if (!is.na(my_ending_stop) && !is.na(length(real_times1)) && my_ending_stop>length(real_times1))
           my_ending_stop < length(real_times1)
         }
        }
        if(real_times_df$bus[i]==1 && freqs[jl]>0 && !is.na(my_ending_stop) && !is.na(my_starting_stop) && length(real_times1[my_ending_stop]  real_times1[my_starting_stop])>0 && !is.na(real_times1[my_ending_stop]  real_times1[my_starting_stop]))
        {
         for(zz in c(1:freqs[jl]))
         {
          if(npass_stat1==sample_pass_capacity1)
           ivts < c(ivts,rep(0,sample_pass_capacity1))
          sample_pass_capacity1 < 2*sample_pass_capacity1
          npass_stat1 < npass_stat1+1

          ivts[npass_stat1] < real_times1[my_ending_stop]  real_times1[my_starting_stop]

          if((real_times1[my_ending_stop]  real_times1[my_starting_stop])<0)
           npass_stat1 < npass_stat1 1
         }
        }
       }
      }
     }
     else
      exits_in_stops[idx_bus] < 0


     if (idx_bus>1)
      my_bus[idx_bus] < my_bus[idx_bus 1]  exits_in_stops[idx_bus]
     else
      my_bus[idx_bus] < pass_in_stops[idx_bus]

     #actions deployment...
     if(select_action==3 && stops_action[1]==idx_bus && real_times_df$bus[i]==1)
     {
      print(sprintf("stop_%d_was_skipped_by_bus_1!!!",idx_bus))

      pass_in_stops[idx_bus] < 0

```

```
    }

    #filled bus case on boardings
    free_space<bus_capacity my_bus[idx_bus]
    if (pass_in_stops[idx_bus]>free_space)
    {
     pass_in_stops[idx_bus]<free_space
     passengers_stops2$PASSENGERS_WAITING[sel.stops]<passengers_stops2$PASSENGERS_WAITING[sel.stops] free_space
     my_bus[idx_bus]<bus_capacity
    }
    else
    {
     #base case on boardings....
     my_bus[idx_bus]<my_bus[idx_bus]+pass_in_stops[idx_bus]
     passengers_stops2$PASSENGERS_WAITING[sel.stops]<0
    }

    #passenger waiting times generation
    if(real_times_df$bus[i]==2 && !is.na(real_times2[idx_bus] real_times1[idx_bus]) && !is.na(pass_in_stops[idx_bus]) && (real_times2[idx_bus]> real_times1[idx_bus]))
    {
     if(pass_in_stops[idx_bus]>0)
     {
      arrival_times_pass<round(rexp(pass_in_stops[idx_bus]+1,pass_in_stops[idx_bus])*abs(real_times2[idx_bus] real_times1[idx_bus]))
      if (sum(arrival_times_pass)>abs(real_times2[idx_bus] real_times1[idx_bus]))
      {
       arrival_times_pass<arrival_times_pass*abs(real_times2[idx_bus] real_times1[idx_bus])/sum(arrival_times_pass)
      }
      arrival_times_pass<rev(rev(arrival_times_pass)[ 1])

      if(length(arrival_times_pass)>0)
      {

       su<abs(real_times2[idx_bus] real_times1[idx_bus])
       if(length(arrival_times_pass)>1)
       {
        for(zz in c(2:(length(arrival_times_pass))))
         arrival_times_pass[zz]<arrival_times_pass[zz]+arrival_times_pass[zz 1]

        arrival_times_pass<round(su arrival_times_pass)

       }


      }
```

```
        if(length(arrival_times_pass)>0)
        {
         for (jl in c(1:pass_in_stops[idx_bus]))
         {
          if(!is.na(arrival_times_pass[jl]))
          {

           if(npass_stat2==sample_pass_capacity2)
            wts<c(wts,rep(0,sample_pass_capacity2))
           npass_stat2<npass_stat2+1


           wts[npass_stat2]<arrival_times_pass[jl]
           if(wts[npass_stat2]>3600*2)
            npass_stat2<npass_stat2 1
          }
         }

        }
       }
      }


      for (j in c(1:length(common_stops)))
      {
       new_stops<common_stops[j]
       sel.stops<which(passengers_stops2$STOP==new_stops &
           passengers_stops2$ACTIV==1)
       if (length(sel.stops)>0)
       {
        delta<max(0,real_times_df$real_times[i] passengers_
            stops2$LAST_UPDATED[sel.stops])
        if(is.na(delta)|| is.na(max_frequency_pass))
        {
         print("NA3")
        }
        if(delta<(max_frequency_pass*60))
         passengers_stops2$PASSENGERS_WAITING[sel.stops]<min(
             passengers_stops2$PASSENGERS_WAITING[sel.stops]+
             round(delta%/%weights_demand[idx_bus]*(factores_
             procura[idx_bus])),min(round(max_passengers_stop*
             factores_procura[idx_bus]),max_passengers_stop))
        else
         passengers_stops2$PASSENGERS_WAITING[sel.stops]<min(
             round(new_stops_demand*factores_procura[idx_bus]),
             min(round(max_passengers_stop*factores_procura[idx_
             bus]),max_passengers_stop))

        if (round(delta%/%weights_demand[idx_bus]*(factores_
            procura[idx_bus]/2))>30 || round(delta%/%weights_
            demand[idx_bus]*(factores_procura[idx_bus]/2))<0)
        {
         print("ALARME3")
```

```

          }

          passengers_stops2$LAST_UPDATED[sel.stops] < real_times_df
              $real_times[i]

        }
      }
      current_time < current_time+delta

      if(real_times_df$bus[i]==1)
      {
        bus1 < my_bus
        pass_in_stops1 < pass_in_stops
        exits_in_stops1 < exits_in_stops
      }
      else
      {
        bus2 < my_bus
        pass_in_stops2 < pass_in_stops
        exits_in_stops2 < exits_in_stops
      }

    }
    exits_in_stops1[idx_bus] < exits_in_stops1[idx_bus]+(sum(
        pass_in_stops1[1:idx_bus]) sum(exits_in_stops1[1:idx_
        bus]))


    prev.pass_in_stops1 < pass_in_stops1[1:idx_bus]

    #plotting
    if (PLOT_DEMAND==TRUE && select_action>0 )
    {
      if(select_action==1)
        actionstr < "NONE"
      if(select_action==2)
        actionstr < "BUS_HOLDING"
      if(select_action==3)
        actionstr < "STOP_SKIPPING"
      demand_chart(as.numeric(year_date),as.numeric(month_date),
          as.numeric(day_date),start_weekday,current_time%/%
          3600,round(((current_time/3600) (current_time%/%3600))
          *60),sprintf("%s_%s",line,way),bus1[1:idx_bus],pass_in
          _stops1[1:idx_bus],exits_in_stops1[1:idx_bus],bus_
          capacity,PLOT_DEMAND,actionstr,stops_action[1])
    }
    #Total In Vehicle Time
    LTT < real_times1[2:idx_bus] real_times1[1:(idx_bus 1)]
    IVT < sum(LTT*bus1[1:(idx_bus 1)])
    if(is.na(IVT))
      IVT < 0
    TIVT < IVT+TIVT

```

```
    B1<sum(pass_in_stops1)
    if(is.na(B1))
     B1<0
    TB1<TB1+B1

    exits_in_stops2[idx_bus] < exits_in_stops2[idx_bus]+(sum(pass_in_stops2[1:idx_bus])  sum(exits_in_stops2[1:idx_bus]))
    prev.pass_in_stops2 < pass_in_stops2[1:idx_bus]


    #Total Waiting Time
    stop_hdw< real_times2  real_times1
    PWT<sum(stop_hdw/2*pass_in_stops2)
    if(is.na(PWT))
     PWT< 0


    TWT<PWT+TWT

    B2<sum(pass_in_stops2)
    if(is.na(B2))
     B2<0
    TB2<TB2+B2

    passengers_stops2 < passengers_stops
    real_headways<abs(real_times2  real_times1)
   }
   else
   {
    pre1 < 0
    pre2 < 100000
    real_headways< 100000
   }
   #BB detection
   headways<abs(pre2  pre1)
   print("Common_stops:")
   print(common_stops)

   print("Headways:")

   bunching_stops < which(headways<bunching_th)
   bunching_stops2 < which(headways<(bunching_th_inf))
   bunching_stops3 < which(headways<(bunching_th_sup))
   if(length(bunching_stops)>0)
   {
    idx_stop < bunching_stops[1]
    idx_stop < idx1[idx_stop]
    bunching$STOP[idx] < trip1$ID_PARAGEM_STCP[idx_stop]
    bunching$BUNCHING[idx] < "BUNCHING"
    bunching$PROBABILITY1[idx] <min((min(1,length(bunching_stops3)/9)/2)+(min(1,length(bunching_stops)/6)/3)+(min(1,length(bunching_stops2)/3)/6),1)
   }
```

```
else
{
 bunching$PROBABILITY1[idx] < (min(1,length(bunching_stops3)/9)/2)
}

mode< "incremental"

#BB probabilistic model
if (length(idx1)>10)
{
 if (mode=="offline")
 {
  online_headways< headways
  online_headways[1] < real_headways[1]
  for(i in c(2:length(headways)))
   online_headways[i] < real_headways[i 1]+ headways[i] headways[i 1]

  beta_i < beta
  online_delta_rule < online_headways
  for(i in c(2:length(headways)))
  {
   online_delta_rule[i] < online_delta_rule[i]+((real_headways[i 1] online_delta_rule[i 1])*beta_i)
   if (abs(online_delta_rule[i] real_headways[i])>abs(online_headways[i] real_headways[i]))
    beta_i < beta_i*(1 0.1)
   else
    if (abs(abs(online_delta_rule[i] real_headways[i]) abs(online_headways[i] real_headways[i]))>10)
     beta_i < beta_i*(1+0.1)
   beta_i < min(max(0.005,beta_i),0.3)
  }
  headways_up< online_delta_rule+median(recent_maes)
  headways_down< online_delta_rule median(recent_maes)
  bunching_prob< rep(bunching_th,length(online_delta_rule))


  sel_idx < which(bunching_prob>=headways_up)
  sel_idx2 < which(bunching_prob<headways_up & bunching_prob>headways_down)
  sel_idx3 < which(bunching_prob<=headways_down)
  if (length(sel_idx)>0)
   bunching_prob[sel_idx] < 1

  if (length(sel_idx3)>0)
   bunching_prob[sel_idx3] < 0
  bunching_prob[sel_idx2] < abs(bunching_prob[sel_idx2] headways_down[sel_idx2])/(2*median(recent_maes))

  bunching$PROBABILITY2[idx] < max(bunching_prob)

  for(i in c(1:(sliding_window_MAE_size 1)))
```

```
      recent_maes[i] < recent_maes[i+1]

    dif_headways < real_headways  online_delta_rule

    recent_maes[length(recent_maes)] < mean(abs(dif_headways))

    bunching$headway_MAE_delta_rule_online[idx] < mean(abs(real_headways  online_headways))
    bunching$headway_MAE_inter_trip[idx] < mean(abs(real_headways  headways))
    bunching$online_delta_rule < mean(abs(real_headways  online_delta_rule))
    bunching$nstops[idx] < length(headways)
   }
   else
   {
    online_headways < headways
    online_headways[1] < real_headways[1]
    for(i in c(2:length(headways)))
     online_headways[i] < real_headways[i 1]+ headways[i]  headways[i 1]

    beta_i < beta
    online_delta_rule < online_headways
    for(i in c(2:length(headways)))
    {
     online_delta_rule[i] < online_delta_rule[i]+((real_headways[i 1]  online_delta_rule[i 1])*beta_i)
     if(is.na(online_delta_rule[i]))
      idx1 < 1
     else
     {
      #NA corrections due to missing values
      if (!is.na(real_headways[i]) && abs(online_delta_rule[i]  real_headways[i])>abs(online_headways[i]  real_headways[i]))
       beta_i < beta_i*(1 0.1)
      else
       if (!is.na(real_headways[i]) && abs(abs(online_delta_rule[i]  real_headways[i])  abs(online_headways[i]  real_headways[i]))>10)
        beta_i < beta_i*(1+0.1)
      beta_i < min(max(0.005,beta_i),0.3)
     }
    }

    nparagens < length(idx1)
    predicted_headways < headways
    predicted_headways[1] < real_headways[1]
    stop_scores < (c(1:nparagens) 1)/nparagens
    np < 2
    bunching_score < 0
    score_th < 0.7

```

```
while (np <=nparagens && max(bunching_score)<=score_th &&
    length(idx1)>10)
{
 print(sprintf("trip:%d, stop %d",idx,np))
 predicted_headways[np] < online_delta_rule[np]

 print("paragens")
 print(c((np+1):nparagens))

 for (i in c((np+1):nparagens))
 {
  predicted_headways[i] < online_delta_rule[i 1]+headways[i
      ] headways[i 1]

 }

 #residual's array
 errors < abs(predicted_headways[c(np:nparagens)] real_
     headways[c(np:nparagens)])
 if (length(is.na(errors))>0)
 {
  sel.nas < which(is.na(errors))
  errors[sel.nas] < 0
 }
 erro_paragem[1:length(errors)] < erro_paragem[1:length(
     errors)]+errors
 n_erros_paragem[1:length(errors)] < n_erros_paragem[1:
     length(errors)]+1

 confidence < sqrt((erro_paragem[1:length(errors)]/n_erros_
     paragem[1:length(errors)])/rep(my_freq,length(errors)
     ))
 idx_zeros < which(confidence<0)
 idx_ones < which(confidence>1)
 if (length(idx_zeros)>0)
  confidence[idx_zeros] < 0
 if (length(idx_ones)>0)
  confidence[idx_ones] < 1
 confidence < 1 confidence (0.15*stop_scores[1:length(
     confidence)])

 errors < (erro_paragem[1:length(errors)]/n_erros_paragem
     [1:length(errors)])

 projections_up < predicted_headways[c(np:nparagens)]+
     errors
 projections_down < predicted_headways[c(np:nparagens)]
     errors

 bunching_prob < rep(bunching_th,length(projections_up))

 sel_idx < which(bunching_prob>=projections_up)
 sel_idx2 < which(bunching_prob<projections_up & bunching_
     prob>projections_down)
```

```
    sel_idx3 < which(bunching_prob<=projections_down)
    if (length(sel_idx)>0)
     bunching_prob[sel_idx] < 1

    if (length(sel_idx3)>0)
     bunching_prob[sel_idx3] < 0
    if (length(sel_idx2)>0)
     bunching_prob[sel_idx2] < abs(bunching_prob[sel_idx2] projections_down[sel_idx2])/(2*errors[sel_idx2])


    sel_real < which(real_headways<bunching_th)
    if(length(sel_real)>1)
     sel_real < sel_real[1]

    #calculus of bunching score
    bunching_scores < bunching_prob
    if (length(bunching_scores)>1)
    {
     num_scores < max(round((1 stop_scores[np])*3),1)
     bunching_score < mean(rev(sort(bunching_scores))[1:num_scores])
    }
    else
     bunching_score < bunching_scores

    #BB threshold for bunching schore
    score_th < 0.3+((my_freq%/%(3*120))*0.1)

    if(bunching_score>score_th)
    {

     sel < which(bunching_scores==max(bunching_scores))
     if(length(sel)>1)
      sel < sample(sel)[1]
     sel_relative < sel
     sel < sel+np 1
     sel_action < round(rlnorm(1,np+2,log(max(2,0.2*length(c(np+1):sel)))))
     if (sel_action>=sel || sel_action<3 || sel_action<np)
      sel_action < np+1

     if (sel_action>sel)
      sel_action < sel

    if (select_action==0)
     {
      if(DO_ACTIONS=="ACTIONS")
      {
       prob_bus_holding < bunching_score

       if(prob_bus_holding==1)
       {
```

```
      prob_stop_skipping < min(1,max(0,rnorm(1,prob_bus_holding,0.1)))
      prob_bus_holding < min(1,max(0,rnorm(1,prob_bus_holding*0.5,0.1)))
     }
     else
      prob_stop_skipping < 0

     print(sprintf("bus holding likelihood: %f",prob_bus_holding))

     print(sprintf("stop skipping likelihood: %f",prob_stop_skipping))


     if(max(prob_bus_holding,prob_stop_skipping)>prob_th_min)
     {
      #select one of the actions
      #2 bus holding
      #3 stop skipping
      if(prob_bus_holding>=prob_stop_skipping)
      {
       select_action < 2
       amount < (round(abs(projections_down[sel_relative]bunching_th)*(1+0.1))%/%minimum_holding_time+1)*minimum_holding_time
       nstops_action < max(1,min(amount%/%minimum_holding_time,(sel  sel_action+1)))

       amount_spread < rep(minimum_holding_time,nstops_action)

       idx_amount < 1
       while(sum(amount_spread)<amount)
       {
        if(idx_amount>length(amount_spread))
         idx_amount < 1
        amount_spread[idx_amount] < amount_spread[idx_amount]+minimum_holding_time
        idx_amount < idx_amount+1
       }

       print(sprintf("Bus Holding! To add the following seconds on the dwell time of stops:"))
       stops_action < c(sel_action:(sel_action+nstops_action 1))
       print(stops_action)
       print(amount_spread)

       if(length(amount_spread)>1)
       {
        for(zz in c(2:length(amount_spread)))
```

```
                amount_spread[zz] < amount_spread[zz 1]+amount_spread[zz]
              }
              if(rev(stops_action)[1]<nparagens)
              {
                amount_spread < c(amount_spread,rep(rev(amount_spread)[1],nparagens rev(stops_action)[1]))
                stops_action < c(stops_action,c((rev(stops_action)[1]+1):nparagens))
              }
              trip_action < 2
            }
            else
            {
              select_action < 3

              amount_spread <max(90,pass_in_stops1[sel_action]*time_for_boarding_per_passenger+min_dwell_time)*1
              if(is.na(amount_spread))
                amount_spread <  30
              print(sprintf("Stop Skipping! To set the dwell time on stop %d on (%d seconds)",sel_action,amount_spread))
              stops_action < sel_action
              amount < amount_spread
              if(sel_action<nparagens)
              {
                amount_spread < c(amount_spread,rep(amount_spread,nparagens  sel_action))
                stops_action < c(stops_action,c((rev(stops_action)[1]+1):nparagens))
              }
              trip_action < 1

            }
          }
          else
          {
            select_action < 1
            stops_action < 0
            amount < 0
          }
        }
        else
        {
          select_action < 1
          stops_action < 0
          print("no action")
          amount < 0
        }
        print("decision")
        print(select_action)
      }
```

```
    else
    {
      if (select_action==1)
        bunching$ACTION[idx] < "NONE"
      if (select_action==2)
        bunching$ACTION[idx] < "BUS_HOLDING"
      if (select_action==3)
        bunching$ACTION[idx] < "STOP_SKIPPING"

      select_action < 0
      cons < 0

      bunching$time_amount[idx] < amount
      bunching$RESULTED[idx] < 0
    }
    print(sprintf("Bunching Predicted on the stop %d. 
        Horizon: %d stops. Action recomended on the stop %d 
        (at most!)",sel,sel np,sel_action))
    if (length(sel_real)==1)
    {
      print(sprintf("CORRECT! Bunching on the stop %d",sel_
          real))
      bunching$stop_ocurred[idx] < sel_real
    }
    else
    {
      print("WRONG: there is no bunching")

    }

    bunching$BUNCHING_ONLINE[idx] < "BUNCHING"
    bunching$stop_predicted[idx] < sel
    bunching$stop_action[idx] < sel_action
    bunching$prediction_stop[idx] < np
    bunching$nstops[idx] < length(headways)
  }
  else
  {
    cons < 0

    if (length(sel_real)>=1 && sel_real<=np)
    {
      print(sprintf("UNDETECTED_BUNCHING_OCCURRED_ON stop %d"
          ,np))
      bunching_score < 1
      bunching$stop_ocurred[idx] < sel_real
      bunching$stop_predicted[idx] < sel_real
      bunching$stop_action[idx] < sel_real
      bunching$prediction_stop[idx] < sel_real
      bunching$nstops[idx] < length(headways)
      select_action < 0
    }
  }

```

```
        np < np+1
      }

      if(length(sel_real)==0 && select_action>0)
      {
        if (select_action==1)
         bunching$ACTION[idx] < "NONE"
        if (select_action==2)
         bunching$ACTION[idx] < "BUS_HOLDING"
        if (select_action==3)
         bunching$ACTION[idx] < "STOP_SKIPPING"
        select_action < 0
        cons < 0

        bunching$time_amount[idx] < amount

        bunching$RESULTED[idx] < 1

        select_action < 0
      }
     }

    }
    else
    {
     bunching$headway_MAE_delta_rule_online[idx] <  1
     bunching$headway_MAE_inter_trip[idx] <  1
     bunching$online_delta_rule[idx] <  1
     bunching$nstops[idx] < length(headways)
    }
    if (bunching$PROBABILITY2[idx]>0.75)
     bunching$BUNCHING[idx] < "BUNCHING"
    else
     bunching$BUNCHING[idx] < "NO_BUNCHING"
    if (length(idx1)<=10)
    {
     bunching$BUNCHING[idx] < "N/A"
     bunching$BUNCHING_ONLINE[idx] < "N/A"
     TWT<TWT PWT
     TIVT < TIVT  IVT
     TB1 < TB1  B1
     TB2 < TB2  B2
     npass_stat2 < old_npass_stat2
     npass_stat1 < old_npass_stat1
    }
    else
    {
     bunching$TWT[idx] < round(sum(wts[1:npass_stat2])/npass_
         stat2)
     bunching$TIVT[idx] < round(sum(ivts[1:npass_stat1])/npass_
         stat1)
     bunching$TB2[idx] < npass_stat2
     bunching$TB1[idx] < npass_stat1

```

```
  if(npass_stat2%/%(plot_periodicity+chart_sample_periodicity)>0)
  {
   plot_periodicity < plot_periodicity+chart_sample_periodicity

   histogram_equal_width(as.numeric(year_date),as.numeric(month_date),as.numeric(day_date),sprintf("%s_%s",line,way),wts[1:npass_stat2],"Waiting_Time",DO_ACTIONS)
   print(length(which(wts[1:npass_stat2]<60))/npass_stat2)
  }

  sel.nas < which(is.na(real_headways))
  if(length(sel.nas)>0)
   real_headways < real_headways[ sel.nas]

  if(npass_stat3+length(real_headways)>sample_pass_capacity3)
  {
   headways_array < c(headways_array,rep(0,sample_pass_capacity3))
   sample_pass_capacity3 < 2*sample_pass_capacity3
  }

  for(zz in c(1:length(real_headways)))
  {
   npass_stat3 < npass_stat3+1
   headways_array[npass_stat3] < real_headways[zz]
   if(headways_array[npass_stat3]<0 || headways_array[npass_stat3]>(3600*1.5))
    npass_stat3 < npass_stat3 1
  }

  if(npass_stat3%/%(plot_periodicity2+chart_sample_periodicity)>0)
  {
   plot_periodicity2 < plot_periodicity2+chart_sample_periodicity
   histogram_equal_width(as.numeric(year_date),as.numeric(month_date),as.numeric(day_date),sprintf("%s_%s",line,way),headways_array[1:npass_stat3],"Headway",DO_ACTIONS)
  }

 }
 print("Result:")
 cur_time < round(as.numeric(proc.time()[3]) start_time)
 avg_time_trip < (cur_time)/(idx)
 hours_time < cur_time%/%3600
 cur_time < cur_time (hours_time*3600)
 minutes_time < cur_time%/%60
 cur_time < cur_time (minutes_time*60)

 expected_time_to_finish < round(avg_time_trip)*(time idx)
 hours_time2 < expected_time_to_finish%/%3600
```

```
    expected_time_to_finish < expected_time_to_finish  (hours_time2*3600)
    minutes_time2 < expected_time_to_finish%/%60
    expected_time_to_finish < expected_time_to_finish  (minutes_time2*60)

  }
  filename=sprintf("LINE%d_S%d_bunching_predicted_sliding_delta_online_v3_freq%d_%s.csv",line,way,round(freq_BB_ratio*100),DO_ACTIONS)
  write.csv2(bunching,filename)
}
```

Appendix C

Source Code of Taxi Demand Prediction

```
#real time taxi demand prediction (pick up quantity)
prediction < function(myPart, testeid, tseries30, tseries5, BEG_LEARN, END_LEARN, alpha=0.4, gamma=8, theta=2, H=8)
{
 #preprocessing and libraries loading
   libraries2()
   curDate < incrementa_data(END_LEARN, "/")
   #select the stops correspondent to the part
   stops < unique(tseries30$Praca)
   stops < stops[ten_fold_interval(length(stops), myPart)]
   len < length(tseries5$Data)
   DATA_FIM < tseries5$Data[len]
   DATA_FIM < substr(DATA_FIM, 1, 10)

   len_arima=48*7*theta
   models < as.data.frame(matrix( 1, length(stops), 4))
   models[,1] < stops
   names(models) < c("stand", "p", "d", "q")

   weight_set < calcula_pesos(alpha, gamma, 0)

   len < (24*5)*30*12*100
   measures < matrix(" 1", len, 8)

   step < 5

   idx < 0

   DATA_FIM < decrementa_data(DATA_FIM, "/")

    #for every dates in the stream
 while (curDate<=DATA_FIM)
 {
   diaSemana < date_to_dayweek(curDate)
   for(st in stops)
```

```
    {
 nprev < 0
 query_stops < which(tseries5$Praca==st)
 data_stops5 < tseries5[query_stops,]

 query_stops < which(tseries30$Praca==st)
 data_stops30 < tseries30[query_stops,]
     h1_30 < 0
 m1_30 < 0

 nhm < next_time(h1_30,m1_30,30)
 h2_30 < nhm[1]
 m2_30 < nhm[2]

 nhm < next_time(h2_30,m2_30,30)
 h3_30 < nhm[1]
 m3_30 < nhm[2]


 while(h1_30<24)
 {

        if (h1_30==23 && m1_30==30)
        {
         ultimo < which(tseries5$Data==incrementa_data(curDate,"/") & tseries5$Praca==st)
         ultimo5 < tseries5[ultimo,]
         ultimo5 < ultimo5[1:6,]
        }
        deslizamentos < 0

        h1_5 < h1_30
        m1_5 < m1_30

        #30 minutes histogram (first level)
        query_serie30 < which( ( ((data_stops30$Hora<h2_30) | (data_stops30$Hora==h2_30 & data_stops30$Minutos<m2_30)) & data_stops30$Data==curDate ) | (data_stops30$Data<curDate))
        serie30 < data_stops30[query_serie30 ,]

        #5 minutes histogram (second level)
        query_serie5 < which(data_stops5$Data<incrementa_data(curDate,"/"))
        serie5 < data_stops5[query_serie5 ,]

        #additive histogram's scrolling
        while(deslizamentos<6)
        {
            series < serie30
            if(deslizamentos>0)
            {
             desliza_subtrai_idx < which(serie5$Hora==h1_5 & serie5$Minutos==m1_5 & serie5$Data==curDate)
```

```
      desliza_subtrai_idx < rev(seq(from=desliza_subtrai_idx[1], to=1,by= 6))
      desliza_soma_idx < desliza_subtrai_idx+6

      serie_subtrai5 < serie5$numEventos[desliza_subtrai_idx]
      print(length(desliza_subtrai_idx))
      if (h1_30==23 && m1_30==30)
      {
        serie_soma5< serie5$numEventos[desliza_subtrai_idx[2:length(desliza_subtrai_idx)]]
        serie_soma5<c(serie_soma5, ultimo5$numEventos[deslizamentos])
      }
      else
      {
        serie_soma5< serie5$numEventos[desliza_soma_idx]
      }

      series$numEventos< series$numEventos  serie_subtrai5+serie_soma5
     m1_5<m1_5+5

    }

    #real number of pick ups
    mylen< length(series$numEventos)
    contagem_real < series$numEventos[mylen]
    mylen< mylen  1
    serie_arima < series$numEventos[(mylen  len_arima+1):mylen]
    if (models$p[models$stand==st]== 1 || (h1_30==3 && m1_5==0))
    {

     #arima's model computation
     res < calculaModeloArima(serie_arima)
     models$p[models$stand==st] < res[1]
     models$d[models$stand==st] < res[2]
     models$q[models$stand==st] < res[3]

     msg< sprintf("Novo_Modelo:_%d_%d_%d", res[1], res[2], res[3])
     print(msg)

    }

    model=c(models$p[models$stand==st], models$d[models$stand==st], models$q[models$stand==st])

    query_media < which(series$DiaSemana==diaSemana & series$Data<curDate & series$Minutos==m1_30 & series$Hora==h1_30)
    serie_avg < series$numEventos[query_media]
```

```

      valor_real < contagem_real

      #Arima
      previsao_arima < previsaoArima(model,  serie_arima)
      #Time varying poisson model
      previsao_media < mean(serie_avg)
      #weighted Time varying poisson model
      mylen < length(serie_avg)
      previsao_media_pesada < calcula_wpoisson(serie_avg[(
          mylen gamma+1):mylen], weight_set)

      nprev < nprev+1
      if ((nprev 1)<H)
      {
       previsao < round(previsao_media_pesada)
      }
      else
      {
       idx_media_pesada < which(measures[,7]=="WPoisson")
       idx_media < which(measures[,7]=="Media")
       idx_arima < which(measures[,7]=="Arima")
       idx_real < which(measures[,7]=="Real")

       len_erro < length(idx_media_pesada)
       idx_media_pesada < idx_media_pesada[(len_erro H+1):
           len_erro]
       idx_media < idx_media[(len_erro H+1):len_erro]
       idx_arima < idx_arima[(len_erro H+1):len_erro]
       idx_real < idx_real[(len_erro H+1):len_erro]

       past_media_pesada < as.numeric(measures[idx_media_
           pesada,8])
       past_media < as.numeric(measures[idx_media,8])
       past_arima < as.numeric(measures[idx_arima,8])
       past_real < as.numeric(measures[idx_real,8])

       errA < 1 sMAPE_agg(past_arima, past_real)
       errM < 1 sMAPE_agg(past_media, past_real)
       errMP < 1 sMAPE_agg(past_media_pesada, past_real)

       if (previsao_arima==1000)
        errA < 0

       previsao < round(((errA*previsao_arima)+(errM*
           previsao_media)+(errMP*previsao_media_pesada))/
           (errA+errM+errMP))
      }

      print("")
      print("")
      print(st)
      print(curDate)
      print(h1_30)
```

```
            print(m1_5)
            print(previsao)
            msg<sprintf("Praca:%d␣Data:␣%s␣Hora:␣%2d:%2d␣Eventos␣Previstos:␣%d",st,curDate,h1_30,m1_5,as.numeric(previsao))
            print(msg)
            print("")

            print(c("Index","Praca","DiaSemana","Data","Hora","Minutos","Algoritmo","numEventos"))

            idx<idx+1

            print(c(idx,st,diaSemana,curDate,h1_30,m1_5,"Media",round(previsao_media)))
            measures[idx,]<c(idx,st,diaSemana,curDate,h1_30,m1_5,"Media",round(previsao_media))

            idx<idx+1

            print(c(idx,st,diaSemana,curDate,h1_30,m1_5,"Arima",round(previsao_arima)))
            measures[idx,]<c(idx,st,diaSemana,curDate,h1_30,m1_5,"Arima",round(previsao_arima))

            idx<idx+1

            print(c(idx,st,diaSemana,curDate,h1_30,m1_5,"WPoisson",round(previsao_media_pesada)))
            measures[idx,]<c(idx,st,diaSemana,curDate,h1_30,m1_5,"WPoisson",round(previsao_media_pesada))

            idx<idx+1

            print(c(idx,st,diaSemana,curDate,h1_30,m1_5,"Previsao",round(previsao)))
            measures[idx,]<c(idx,st,diaSemana,curDate,h1_30,m1_5,"Previsao",round(previsao))

            idx<idx+1

            print(c(idx,st,diaSemana,curDate,h1_5,m1_5,"Real",round(contagem_real)))
            measures[idx,]<c(idx,st,diaSemana,curDate,h1_30,m1_5,"Real",round(contagem_real))

            deslizamentos<deslizamentos+1

        }

        h1_30<h2_30
        m1_30<m2_30

        nhm<next_time(h1_30,m1_30,30)
```

```
        h2_30 < nhm[1]
        m2_30 < nhm[2]

        nhm < next_time(h2_30,m2_30,30)
        h3_30 < nhm[1]
        m3_30 < nhm[2]

      }

    }
    curDate < incrementa_data(curDate,"/")

    measures2 < measures[1:idx,]
    measures2 < as.data.frame(measures2)
    names(measures2) < c("Index","Praca","DiaSemana","Data","Hora","Minutos","Algoritmo","numEventos")
    filename < sprintf("teste%d_part%d.csv",testeid,myPart)
    write.csv2(measures2,filename)
  }

  measures < measures[1:idx,]
  measures < as.data.frame(measures)
  names(measures) < c("Index","Praca","DiaSemana","Data","Hora","Minutos","Algoritmo","numEventos")
  filename < sprintf("teste%d_part%d.csv",testeid,myPart)
  write.csv2(measures,filename)
  return(measures)
}
```

Appendix D

Source Code of O-D Matrix Estimation

```
#preprocessing dataset
generate_trips_dataset<function(ds,spatial_data_set,startday=0,endday="2012/05/29")
{
 vars<c("start_day","starting_hour","starting_minutes","weekday","lat1","lon1","lat2","lon2","distance","travel_time")
 dataset<data.frame(taxi=ds$taxi,type=ds$type,timestamp=ds$timestamp,lat=ds$latitude,lon=ds$longitude)
 dataset$timestamp<as.character(dataset$timestamp)

 if(is.character(startday))
 {

  dataset<dataset[which(dataset$timestamp>=startday),]

 }
 dataset<dataset[ which(dataset$timestamp>=endday),]


 my_search<which(substr(dataset$timestamp,11,11)!=" ")
 dataset$timestamp[my_search]<sprintf("%s %s",substr(dataset$timestamp[my_search],1,10),substr(dataset$timestamp[my_search],11,length(dataset$timestamp[my_search])))

 dataset<dataset[ which(dataset$lat==0),]

 dataset<dataset[ which(dataset$lon==0),]

 dataset$type<as.character(dataset$type)
 dataset<dataset[ which(dataset$type=="assign"),]

 print("matrix_ready")
 m<matrix("0",round(length(dataset$timestamp)/2),length(vars))
```

```
taxi < unique(dataset$taxi)

i < 1
for(t in taxi)
{
 ds_taxi < dataset[which(dataset$taxi==t),]
 idx_busy < which(ds_taxi$type=="busy")
 print(sprintf("taxi: %d, potential trips: %d",t,round(length(idx_busy))))

 if (length(idx_busy)>0)
 {
  for(idx in idx_busy)
  {
   idx_free < idx+1

   while(!is.na(ds_taxi$type[idx_free]) && ds_taxi$type[idx_free]!="free")
    idx_free < idx_free+1
   if (!is.na(ds_taxi$type[idx_free]))
   {
    start_day < substr(as.character(ds_taxi$timestamp[idx]),1,10)
    print(ds_taxi$timestamp[idx])
    print(as.character(ds_taxi$timestamp[idx]))
    print(start_day)

    start_hour < substr(as.character(ds_taxi$timestamp[idx]),12,13)
    print(start_hour)

    starting_minutes < (as.numeric(substr(as.character(ds_taxi$timestamp[idx]),12,13))*60)+as.numeric(substr(as.character(ds_taxi$timestamp[idx]),15,16))
    print(starting_minutes)

    dow < DAY_OF_WEEK(as.character(ds_taxi$timestamp[idx]))
    print(dow)

    lat1 < ds_taxi$lat[idx]
    lon1 < ds_taxi$lon[idx]
    lat2 < ds_taxi$lat[idx_free]
    lon2 < ds_taxi$lon[idx_free]

    distance < HaversineDistanceObstacules(ds_taxi$lat[idx],ds_taxi$lon[idx],ds_taxi$lat[idx_free],ds_taxi$lon[idx_free])

    travel_time < getDiffSeconds(ds_taxi$timestamp[idx_free],ds_taxi$timestamp[idx])
   m[i,] < c(start_day,start_hour,starting_minutes,dow,lat1,lon1,lat2,lon2,distance,travel_time)

    i < i+1
```

```
      }
    }
  }
 }
 print(i)
m<m[1:(i 1) ,]
m< as.data.frame(m)
names(m) < vars
m$lat1 < as.double(as.character(m$lat1))
m$lon1 < as.double(as.character(m$lon1))
m$lat2 < as.double(as.character(m$lat2))
m$lon2 < as.double(as.character(m$lon2))
m$starting_minutes < as.numeric(as.character(m$starting_minutes))
m$distance < as.double(as.character(m$distance))
m$travel_time < as.numeric(as.character(m$travel_time))
 str(m)
 print(vars)
 myname< sprintf("triptraveltimes_ntrips=%d_v2.csv",i)
 print(myname)
 write.csv2(m,myname)
 return(m)
}

#filtering the data to be used in the experiments based on user defined parameters
preprocessing_dataset < function(ds, radius=12000,center="Porto")
{
  if (center=="Porto")
    center=c(41.153972, 8.61295);

  ds < data.frame(taxi=ds$taxi, type=as.character(ds$type), lat=ds$latitude, lon=ds$longitude, timestamp=as.character(ds$timestamp))

  idx < which(ds$lat==0)
  if (length(idx)>0)
 ds < ds[ idx,]

  idx < which(ds$lon==0)
  if (length(idx)>0)
 ds < ds[ idx,]

  idx < which(is.na(ds$lon))
  if (length(idx)>0)
 ds < ds[ idx,]

  idx < which(ds$taxi==0)
  if (length(idx)>0)
 ds < ds[ idx,]

  idx < which(is.na(ds$taxi))
  if (length(idx)>0)
```

```
 ds< ds[ idx,]

  hs< HaversineDistance(rep(center[1],length(ds[,1])),rep(
      center[2],length(ds[,1])),ds$lat,ds$lon)
  idx< which(hs>radius)
  print(sprintf("%d points farthest than %d meters removed",
      length(idx),radius))
  ds< ds[ idx,]

  ds$timestamp< as.character(ds$timestamp)
  ds$type< as.character(ds$type)

  ds< ds[with(ds,order(timestamp)),]

  return(ds)
}

#compute centroids for naive clustering
generateCentroids< function(centers,k)
{
 largura< seq(from=centers$lat[1],to=centers$lat[3],by=(
     centers$lat[3] centers$lat[1])/k)+(((centers$lat[3]
     centers$lat[1])/k)/2)
 altura< seq(from=centers$lon[1],to=centers$lon[2],by=(centers
     $lon[2] centers$lon[1])/k)+(((centers$lon[2] centers$lon
     [1])/k)/2)
 m< matrix(0,k*k,2)
 id< 1
 for(i in c(1:(k)))
  for(j in c(1:(k)))
  {
   m[id,1]< largura[i]
   m[id,2]< altura[j]
   print(sprintf("%f,%f",largura[i],altura[j]))
   id< id+1
  }
 m< as.data.frame(m)
 names(m)< c("lat","lon")
 return(m)
}

#get cluster for data point on naive clustering
centroidClus< function(lat,lon,centroids)
{
 distance< HaversineDistance(lat,lon,centroids$lat,centroids$
     lon)
 return(which(distance==min(distance)))
}

#naive clustering function
naiveClustering< function(lat,lon,centroids)
{
 len< length(lat)
 clusters< rep(1,len)
```

```
  for(i in 1:len)
  {
    clusters[i] < centroidClus(lat[i],lon[i],centroids)
    print(sprintf("lat:%f, lon:%f, cluster:%d",lat[i],lon[i],
        clusters[i]))
  }

  return(clusters)
}

#initializes half space tree
build.tree < function(maxDim)
{
  vars < c("IDnode","node_type","condition_type","operator","
      value","left","right")
  m< matrix("0",maxDim,length(vars))
  print ("building new tree...")
  print (sprintf("Maximum number of nodes: %d...",maxDim))

  m[1,] < c("1","condition","lat","greater","67.098","3","4")
  m[2,] < c("2","cluster","lon","lower"," 47.098","2","NA")

  m< as.data.frame(m)
  names(m) < vars
  m$IDnode < as.numeric(m$IDnode)
  m$node_type < as.factor(m$node_type)
  m$condition_type < as.factor(m$condition_type)
  m$operator < as.factor(m$operator)
  m$value < as.numeric(m$value)
  m$left < as.numeric(m$left)
  m$right < as.numeric(m$right)

  str(m)
  return(list(m,0,maxDim))
}

#clustering using half space tree
getTreeCluster < function(lat,lon,tree)
{
  node < tree[[1]][1,]

  nchamadas < 0
  while (node$node_type!="cluster")
  {
    if (node$condition_type=="lat")
    {
      if(lat>node$value)
        newnode < node$right
      else
        newnode < node$left
    }
    else
    {
      if(lon>node$value)
```

```
      newnode < node$right
    else
      newnode < node$left
   }

   idxnode < which(tree[[1]]$IDnode==newnode)[1]
   node < tree[[1]][idxnode,]

  }
  return(node$value)
}


#offline clustering
mass_based_clustering < function(ds,N, plotting=TRUE,max.points.
    perc=0.05,optimal.ratio.split=0.1,minimum.density.ratio.
    split=0.1,interest.ratio=0.01,kMax=200,mytree=0,clusters
    =0,ntrips_tosave=5,rect.proportion=3)
{
 library(sp)
 trips < ds
 total.len.trips < length(ds[,1])

    #data reading
 new.ds < data.frame(lat=c(ds$lat1[1:N],ds$lat2[1:N]),lon=c(ds$
     lon1[1:N],ds$lon2[1:N]))
 new.ds$lat[seq(from=1,to=N*2 1,2)] < ds$lat1[1:N]
 new.ds$lon[seq(from=1,to=N*2 1,2)] < ds$lon1[1:N]
 new.ds$lat[seq(from=2,to=N*2,2)] < ds$lat2[1:N]
 new.ds$lon[seq(from=2,to=N*2,2)] < ds$lon2[1:N]
 ds < new.ds

 str(ds)

 N<N*2
 len < length(ds[,1])
 nsample < min(N,5000)
 res < all.grid(ds,4,nsample)[[3]]
 map.center < c(res$lat.center,res$lon.center)

 topleft < c(res$BBOX$ll[1,1],res$BBOX$ll[1,2])
 rightbottom < c(res$BBOX$ur[1,1],res$BBOX$ur[1,2])

 obst < read.obstacules()
 #variable's initialization
 if (!is.data.frame(mytree))
 {
  mytree< build.tree(kMax*5)

  mytree[[1]] < insertNode(mytree[[1]],1,"cluster",NA,NA,1,NA,
      NA)
  mytree[[2]] < 1


```

```
    vars2<c("cluster","mass","lat1","lon1","lat2","lon2","polygonType","polygonID")
   print("Build␣new␣cluster␣structure...")
    cluster<matrix(0,kMax,length(vars2))
    cluster[1,]<c(1,N,topleft[1],topleft[2],rightbottom[1],rightbottom[2],"cutted",1)
    cluster<as.data.frame(cluster)
   names(cluster)<vars2
    cluster$polygonType<as.character(cluster$polygonType)
    cluster$polygonID<as.numeric(as.character(cluster$polygonID))
    for (i in c(1:6))
      cluster[,i]<as.numeric(as.character(cluster[,i]))
    clusters<list(cluster,1,kMax)

    operations<matrix(0,kMax*2,4)
    vars3<c("original","new1","new2","type")
    operations<as.data.frame(operations)
   names(operations)<vars3
    operations<list(operations,0,kMax*2)

    vars4<c("polygonID","lat1","lon1","lat2","lon2","lat3","lon3","lat4","lon4")
    irrPolygons<matrix(0,kMax,length(vars4))

   names(irrPolygons)<vars4
    irrPolygons<as.data.frame(irrPolygons)
    irrPolygons<list(irrPolygons,0,kMax)

    obj<updateIPoly(clusters,1,irrPolygons,getIrregularPolygon(topleft[1],topleft[2],rightbottom[1],rightbottom[2],obst))
    irrPolygons<obj[[2]]
    clusters<obj[[1]]
  }

  iteration<1
  ma<max(clusters[[1]]$mass[1:clusters[[2]]])
  total.clusters<rep(1,N)


  color_pallette_basic<primary.colors(30,3,"FALSE")
  color_pallette_complex<sample(primary.colors(100,5,"FALSE"))
  color_pallette_complex<color_pallette_complex[ which(color_pallette_complex %in% color_pallette_basic)]
  color_pallette<c(color_pallette_basic,sample(color_pallette_complex))

     #offline stage   half space tree
  while(ma>round(max.points.perc*N))
  {
    cluster_to_divide<clusters[[1]]$cluster[which(clusters[[1]]$mass==ma)]
    cluster_to_divide<cluster_to_divide[1]
```

```
idxnode<which(mytree[[1]]$node_type[1:mytree[[2]]]=="
    cluster" & mytree[[1]]$value[1:mytree[[2]]]==cluster_to_
    divide)
IDleaf<mytree[[1]]$IDnode[idxnode]
ds.cluster<ds[which(total.clusters==cluster_to_divide),]

      #obtains the cluster division...which would be the
          median in this stage
opt<find_optimal_division(ds.cluster,clusters[[1]]$mass[
    cluster_to_divide],clusters[[1]]$lat1[cluster_to_divide
    ],clusters[[1]]$lon1[cluster_to_divide],clusters[[1]]$
    lat2[cluster_to_divide],clusters[[1]]$lon2[cluster_to_
    divide],optimal.ratio.split)

      #generates novel clusters
my.poly<getPolygonExtremes(clusters[[1]]$lat1[cluster_to_
    divide],clusters[[1]]$lon1[cluster_to_divide],clusters
    [[1]]$lat2[cluster_to_divide],clusters[[1]]$lon2[cluster
    _to_divide],opt[[1]],opt[[2]])
str(my.poly)
clusters[[1]][cluster_to_divide,]<c(cluster_to_divide,
    clusters[[1]]$mass[cluster_to_divide] opt[[3]],my.poly
    [[1]][1],my.poly[[1]][2],my.poly[[1]][3],my.poly
    [[1]][4],0,0)
obj<updateIPoly(clusters,cluster_to_divide,irrPolygons,
    getIrregularPolygon(my.poly[[1]][1],my.poly[[1]][2],my.
    poly[[1]][3],my.poly[[1]][4],obst))
irrPolygons<obj[[2]]
clusters<obj[[1]]
clusters[[2]] < clusters[[2]]+1
new.cluster < clusters[[2]]
clusters[[1]][new.cluster,]<c(new.cluster,opt[[3]],my.poly
    [[2]][1],my.poly[[2]][2],my.poly[[2]][3],my.poly
    [[2]][4],0,0)
obj<updateIPoly(clusters,new.cluster,irrPolygons,
    getIrregularPolygon(my.poly[[2]][1],my.poly[[2]][2],my.
    poly[[2]][3],my.poly[[2]][4],obst))
irrPolygons<obj[[2]]
clusters<obj[[1]]

operations[[2]] < operations[[2]]+1
operations[[1]][operations[[2]],]<c(cluster_to_divide,
    cluster_to_divide,new.cluster,1)

mytree[[1]] <insertNode(mytree[[1]],IDleaf,"condition",opt
    [[1]],opt[[4]],opt[[2]],mytree[[2]]+1,mytree[[2]]+2)
mytree[[2]] <mytree[[2]]+2

if (mytree[[2]]>mytree[[3]])
  mytree<expanding.tree(mytree)

mytree[[1]] <insertNode(mytree[[1]],mytree[[2]] 1,"cluster",
    NA,NA,cluster_to_divide,NA,NA)
```

```
  mytree[[1]] < insertNode(mytree[[1]], mytree[[2]], "cluster",NA,NA,new.cluster,NA,NA)
  print(mytree[[1]][1:mytree[[2]],])

        #process the clustering
  total.clusters < run_tree(ds, mytree, plotting)

  if (plotting)
  {
   spatial_clustering("mass clustering", clusters[[2]], ds,NULL,zoom=12,total.clusters, list(clusters[[1]][1:clusters[[2]],], color_pallette),FALSE,FALSE,TRUE,TRUE,TRUE,iteration,"pdf",map.center, irrPolygons)
     }

  ma<max(clusters[[1]]$mass[1:clusters[[2]]])
  iteration < iteration+1
 }

    #initialization of novel parameters for the refinement stage
 areas < rep(0, clusters[[2]])
 is.large < areas
 for (cl in c(1:clusters[[2]]))
 {
  areas[cl] < calcAreaIrr(cl, clusters[[1]], irrPolygons[[1]])
  is.large[cl] < is.large.rectangle(rect.proportion, clusters[[1]][cl,])
 }

 my.density < clusters[[1]]$mass[1:clusters[[2]]]/areas
 mean.density < median(my.density[1:clusters[[2]]])
 threshold.density < mean.density*(0.5)
 clusters[[1]] < cbind(clusters[[1]], density=my.density[1:clusters[[3]]])
 clusters[[1]] < cbind(clusters[[1]], interest=rep(1, clusters[[3]]))
 my.clusters < clusters[[1]][1:clusters[[2]],]
 mi < min(my.density[1:clusters[[2]]])

 dense_areas < areas[which(my.clusters$density<threshold.density | is.large==1)]
 maxareas < max(dense_areas)
 idxdens < which(areas==maxareas)

    #refinement cycle
 while((mi<threshold.density && length(idxdens)>0))
 {
  cluster_to_divide < clusters[[1]]$cluster[idxdens]
  idxnode < which(mytree[[1]]$node_type[1:mytree[[2]]]=="cluster" & mytree[[1]]$value[1:mytree[[2]]]==cluster_to_divide)
  IDleaf < mytree[[1]]$IDnode[idxnode]

```

```
    ds.cluster < ds[which(total.clusters==cluster_to_divide),]

        #obtains the best possible division for a given cluster
    opt < find_optimal_division(ds.cluster,clusters[[1]]$mass[cluster_to_divide],clusters[[1]]$lat1[cluster_to_divide],clusters[[1]]$lon1[cluster_to_divide],clusters[[1]]$lat2[cluster_to_divide],clusters[[1]]$lon2[cluster_to_divide],minimum.density.ratio.split,"density")
    old.poly < c(clusters[[1]]$lat1[cluster_to_divide],clusters[[1]]$lon1[cluster_to_divide],clusters[[1]]$lat2[cluster_to_divide],clusters[[1]]$lon2[cluster_to_divide])

    my.poly < getPolygonExtremes(clusters[[1]]$lat1[cluster_to_divide],clusters[[1]]$lon1[cluster_to_divide],clusters[[1]]$lat2[cluster_to_divide],clusters[[1]]$lon2[cluster_to_divide],opt[[1]],opt[[2]])
    str(my.poly)

    clusters[[1]][cluster_to_divide,] < c(cluster_to_divide,clusters[[1]]$mass[cluster_to_divide] opt[[3]],my.poly[[1]][1],my.poly[[1]][2],my.poly[[1]][3],my.poly[[1]][4],0,0,0,1)

    obj < updateIPoly(clusters,cluster_to_divide,irrPolygons,getIrregularPolygon(my.poly[[1]][1],my.poly[[1]][2],my.poly[[1]][3],my.poly[[1]][4],obst))
    irrPolygons < obj[[2]]
    clusters < obj[[1]]

    area1 < calcAreaIrr(cluster_to_divide,clusters[[1]],irrPolygons[[1]])
    clusters[[1]]$density[cluster_to_divide] < clusters[[1]]$mass[cluster_to_divide]/area1
    clusters[[2]] < clusters[[2]]+1
    new.cluster < clusters[[2]]

    clusters[[1]][new.cluster,] < c(new.cluster,opt[[3]],my.poly[[2]][1],my.poly[[2]][2],my.poly[[2]][3],my.poly[[2]][4],0,0,0,1)
    obj < updateIPoly(clusters,new.cluster,irrPolygons,getIrregularPolygon(my.poly[[2]][1],my.poly[[2]][2],my.poly[[2]][3],my.poly[[2]][4],obst))
    irrPolygons < obj[[2]]
    clusters < obj[[1]]

    area2 < calcAreaIrr(new.cluster,clusters[[1]],irrPolygons[[1]])
    clusters[[1]]$density[new.cluster] < clusters[[1]]$mass[new.cluster]/area2

    count.interest < 0

```

```
        #removal of clusters without information from the ROI matrix
    if(clusters[[1]]$mass[cluster_to_divide]<=(interest.ratio*sum(clusters[[1]]$mass[1:clusters[[2]]])))
    {
     clusters[[1]]$interest[cluster_to_divide]<0
     print(sprintf("Cluster %d is no longer of interest...",cluster_to_divide))
     count.interest<count.interest+1
    }
    if(clusters[[1]]$mass[new.cluster]<=(interest.ratio*sum(clusters[[1]]$mass[1:clusters[[2]]])))
    {
     clusters[[1]]$interest[new.cluster]<0
     print(sprintf("Cluster %d is no longer of interest...",new.cluster))
     count.interest<count.interest+1
    }

        #generation of novel refined clusters
    if (count.interest==2)
    {
     clusters[[1]][cluster_to_divide,]<c(cluster_to_divide,clusters[[1]]$mass[cluster_to_divide]+opt[[3]],old.poly[1],old.poly[2],old.poly[3],old.poly[4],0,0,0,2)
     obj<updateIPoly(clusters,cluster_to_divide,irrPolygons,getIrregularPolygon(old.poly[1],old.poly[2],old.poly[3],old.poly[4],obst))
     irrPolygons<obj[[2]]
     clusters<obj[[1]]
     area1<calcAreaIrr(cluster_to_divide,clusters[[1]],irrPolygons[[1]])
     clusters[[1]]$density[cluster_to_divide]<clusters[[1]]$mass[cluster_to_divide]/area1
     clusters[[2]]<clusters[[2]] 1
    }
    else
    {
     operations[[2]]<operations[[2]]+1
     operations[[1]][operations[[2]],]<c(cluster_to_divide,cluster_to_divide,new.cluster,1)
     print(sprintf("Updating tree node %d from leaf to conditional...",IDleaf))
     mytree[[1]]<insertNode(mytree[[1]],IDleaf,"condition",opt[[1]],opt[[4]],opt[[2]],mytree[[2]]+1,mytree[[2]]+2)
     print(sprintf("Creating two new leaves %d and %d",mytree[[2]]+1,mytree[[2]]+2))
     mytree[[2]]<mytree[[2]]+2

     if (mytree[[2]]>mytree[[3]])
      mytree<expanding.tree(mytree)

     mytree[[1]]<insertNode(mytree[[1]],mytree[[2]] 1,"cluster",NA,NA,cluster_to_divide,NA,NA)
```

```
      mytree[[1]] < insertNode(mytree[[1]], mytree[[2]], "cluster", NA,NA,new.cluster,NA,NA)

      total.clusters < run_tree(ds, mytree, plotting)
      print(total.clusters)
    }

    if (plotting)
    {
      spatial_clustering("mass clustering", clusters[[2]], ds,NULL, zoom=12,total.clusters, list(clusters[[1]][1:clusters[[2]],], color_pallette),FALSE,FALSE,TRUE,TRUE,TRUE, iteration,"pdf",map.center, irrPolygons)
    }

    iteration < iteration+1

   my.clusters < clusters[[1]][1:clusters[[2]],]
   print(my.clusters)
   mean.density < median(my.clusters$density[which(my.clusters$interest==1)])

    mi < min(my.clusters$density[which(my.clusters$interest==1)])

    areas < rep(0, clusters[[2]])
    is.large < areas
    for (cl in c(1:clusters[[2]]))
    {
      areas[cl] < calcAreaIrr(cl, clusters[[1]], irrPolygons[[1]])
      is.large[cl] < is.large.rectangle(rect.proportion, clusters[[1]][cl,])
    }
    dense_areas < areas[which((my.clusters$density<threshold.density | is.large==1) & my.clusters$interest==1)]

    maxareas < max(dense_areas)
    idxdens < which(areas==maxareas)
  }
  spatial_clustering("mass clustering", clusters[[2]], ds,NULL, zoom=12,total.clusters, list(clusters[[1]][1:clusters[[2]],], color_pallette),FALSE,FALSE,TRUE,TRUE,TRUE, iteration, c("pdf","final"),map.center, irrPolygons)

 return (list(ds, total.clusters, mytree, clusters, operations, irrPolygons, trips))
}



#online clustering
mass_clustering_stream < function(obj, ds, plot.step=1000, split.test.step=1000, split.ratio=0.05, interest.ratio=0.01, optimal.ratio.split=0.1,minimum.area=1,merge.size.ratio=1.5, rect.proportion=4,minimum.density.ratio.split=0.1,
```

```
    begin.test=200000,end.test=400000,min.perc.mass.travel.
    time=0.75,plotting=TRUE)
{
    #data reading
  my.ds < obj[[1]]
  total.clusters < obj[[2]]
  mytree < obj[[3]]
  clusters < obj[[4]]
  operations < obj[[5]]
  irrPolygons < obj[[6]]
  ds < obj[[7]]

    #preprocessing
  N< 200000
  new.ds < data.frame(lat=c(ds$lat1[1:N],ds$lat2[1:N]),lon=c(ds$
      lon1[1:N],ds$lon2[1:N]))
  new.ds$lat[seq(from=1,to=N*2 1,2)] < ds$lat1[1:N]
  new.ds$lon[seq(from=1,to=N*2 1,2)] < ds$lon1[1:N]
  new.ds$lat[seq(from=2,to=N*2,2)] < ds$lat2[1:N]
  new.ds$lon[seq(from=2,to=N*2,2)] < ds$lon2[1:N]
  ds < new.ds
  N<N*2

  npoints < matrix(0,1000000,2)

  initialN < c(1:length(my.ds[,1]))
  ds < ds[ initialN ,]

  #getting map dimensions
  initialN < max(initialN)
  nsample < min(initialN,5000)
  initialK < clusters[[2]]
  res < all.grid(my.ds,4,nsample)[[3]]
  map.center < c(res$lat.center,res$lon.center)

  #getting global statistics
  total.mass < sum(clusters[[1]]$mass[1:clusters[[2]]])
  max.mass < max(clusters[[1]]$mass[1:clusters[[2]]])

  total.clusters < run_tree(my.ds,mytree,TRUE)

  #getting map obstacules
  obst < read.obstacules()

  #adding mean latitudes and longitudes
  values < rep(0,clusters[[3]])
  clusters[[1]] < cbind(clusters[[1]],meanlat=values)
  clusters[[1]] < cbind(clusters[[1]],meanlon=values)

  clusters[[1]] < cbind(clusters[[1]],initialMassRatio=clusters
      [[1]]$mass/total.mass)

  for(i in c(1:clusters[[2]]))
  {
```

```
   idx < which(total.clusters==i)
   clusters[[1]]$meanlat[i] < mean(my.ds$lat[idx])
   clusters[[1]]$meanlon[i] < mean(my.ds$lon[idx])
 }

 color_pallette_basic < primary.colors(30,3,"FALSE")
 color_pallette_complex < sample(primary.colors(100,5,"FALSE"))
 color_pallette_complex < color_pallette_complex[ which(color_
     pallette_complex %in% color_pallette_basic)]
 color_pallette < c(color_pallette_basic,sample(color_pallette_
     complex))

 len < length(ds[,1])
 count < 0

 len.dataset < length(my.ds[,1])

 np < 1
 for(idx in c(1:len))
 {
   point < ds[idx,]

   cluster < run_tree(point,mytree,TRUE,idx+initialN)
   total.clusters < c(total.clusters,cluster)
   len.dataset < len.dataset+1
   my.ds[len.dataset,] < point

   clusters[[1]]$meanlat[cluster] < clusters[[1]]$meanlat[
       cluster]*clusters[[1]]$mass[cluster]
   clusters[[1]]$meanlat[cluster] < clusters[[1]]$meanlat[
       cluster]+point$lat
   clusters[[1]]$meanlon[cluster] < clusters[[1]]$meanlon[
       cluster]*clusters[[1]]$mass[cluster]
   clusters[[1]]$meanlon[cluster] < clusters[[1]]$meanlon[
       cluster]+point$lon
   clusters[[1]]$mass[cluster] < clusters[[1]]$mass[cluster]+1
   clusters[[1]]$meanlat[cluster] < clusters[[1]]$meanlat[
       cluster]/clusters[[1]]$mass[cluster]
   clusters[[1]]$meanlon[cluster] < clusters[[1]]$meanlon[
       cluster]/clusters[[1]]$mass[cluster]

   area1 < calcAreaIrr(cluster,clusters[[1]],irrPolygons[[1]])
   clusters[[1]]$density[cluster] < clusters[[1]]$mass[cluster]/
       area1



   total.mass < total.mass+1
   count < count+1

         #remove outdated samples
   if (clusters[[1]]$mass[cluster]>max.mass)
   {
     idx < which(total.clusters==cluster)
```

```
   idx < idx[1]

  my.ds <my.ds[ idx,]
   total.clusters < total.clusters[ idx]
   len.dataset < len.dataset 1
  }

  if ((count%%split.test.step)==0)
  {
   changes < 0

   idxsel <which(clusters[[1]]$interest[1:clusters[[2]]]>0 & clusters[[1]]$mass[1:clusters[[2]]]>(split.ratio*total.mass) & calcArea(clusters[[1]]$lat1[1:clusters[[2]]], clusters[[1]]$lon1[1:clusters[[2]]], clusters[[1]]$lat2[1:clusters[[2]]], clusters[[1]]$lon2[1:clusters[[2]]])> minimum.area)
   if (length(idxsel)>0)
   {
    for (idx in idxsel)
    {
     changes < changes+1
     cluster_to_divide < clusters[[1]]$cluster[idx]
     idxnode <which(mytree[[1]]$node_type[1:mytree[[2]]]=="cluster" & mytree[[1]]$value[1:mytree[[2]]]==cluster_to_divide)
     IDleaf < mytree[[1]]$IDnode[idxnode]


     ds.cluster <my.ds[which(total.clusters==cluster_to_divide),]

     mean.density <median(clusters[[1]]$density[1:clusters[[2]]])
     threshold.density <mean.density*(0.5)
     #approximation to the median for getting the ideal split point
     if (clusters[[1]]$density[cluster_to_divide]<threshold.density || is.large.rectangle(rect.proportion, clusters[[1]][cluster_to_divide,])==1)
     {
      opt < find_optimal_division(ds.cluster, clusters[[1]]$mass[cluster_to_divide], clusters[[1]]$lat1[cluster_to_divide], clusters[[1]]$lon1[cluster_to_divide], clusters[[1]]$lat2[cluster_to_divide], clusters[[1]]$lon2[cluster_to_divide], minimum.density.ratio.split, "density")
      if (opt[[5]]>1)
       opt[[5]] < opt[[5]]^ 1

     }
     else
      opt < find_optimal_division(ds.cluster, clusters[[1]]$mass[cluster_to_divide], clusters[[1]]$lat1[cluster_to_
```

```
        divide],clusters[[1]]$lon1[cluster_to_divide],
        clusters[[1]]$lat2[cluster_to_divide],clusters[[1]]$
        lon2[cluster_to_divide],optimal.ratio.split,"none")

  my.poly < getPolygonExtremes(clusters[[1]]$lat1[cluster_to
      _divide],clusters[[1]]$lon1[cluster_to_divide],
      clusters[[1]]$lat2[cluster_to_divide],clusters[[1]]$
      lon2[cluster_to_divide],opt[[1]],opt[[2]])
  str(my.poly)


  mass2 < round(clusters[[1]]$mass[cluster_to_divide]*opt
      [[5]])
  mass1 < clusters[[1]]$mass[cluster_to_divide]  mass2
  points1 < selectPoints(ds.cluster,"lower",opt[[1]],opt
      [[2]])
  points2 < selectPoints(ds.cluster,"greater",opt[[1]],opt
      [[2]])

  print(sprintf("Updating cluster %d to  %d mass points ...
      ",cluster_to_divide,mass1))
  clusters[[1]][cluster_to_divide,] < c(cluster_to_divide,
      mass1,my.poly[[1]][1],my.poly[[1]][2],my.poly
      [[1]][3],my.poly[[1]][4],0,0,0,1,mean(points1$lat),
      mean(points1$lon),mass1/total.mass)

  obj < updateIPoly(clusters,cluster_to_divide,irrPolygons,
      getIrregularPolygon(my.poly[[1]][1],my.poly[[1]][2],
      my.poly[[1]][3],my.poly[[1]][4],obst))
  irrPolygons < obj[[2]]
  clusters < obj[[1]]

  area1 < calcAreaIrr(cluster_to_divide,clusters[[1]],
      irrPolygons[[1]])
  clusters[[1]]$density[cluster_to_divide] < clusters[[1]]$
      mass[cluster_to_divide]/area1
  if (irrPolygons[[2]]==irrPolygons[[3]])
  {
    print(sprintf("Doubling Irregular Polygons list' 
        capacity from %d to %d ...",irrPolygons[[2]],
        irrPolygons[[2]]*2))
    irrPolygons < double.irrPolygon(irrPolygons)
  }

  #creating new cluster
  clusters[[2]] < clusters[[2]]+1
  new.cluster < clusters[[2]]
  print(sprintf("Creating cluster %d with %d mass points...
      ",new.cluster,mass2))
  clusters[[1]][new.cluster,] < c(new.cluster,mass2,my.poly
      [[2]][1],my.poly[[2]][2],my.poly[[2]][3],my.poly
      [[2]][4],0,0,0,1,mean(points2$lat),mean(points2$lon),
      mass2/total.mass)
```

```
      obj < updateIPoly(clusters,new.cluster,irrPolygons,getIrregularPolygon(my.poly[[2]][1],my.poly[[2]][2],my.poly[[2]][3],my.poly[[2]][4],obst))
      irrPolygons < obj[[2]]
      clusters < obj[[1]]
      area2 < calcAreaIrr(new.cluster,clusters[[1]],irrPolygons[[1]])
      clusters[[1]]$density[new.cluster] < clusters[[1]]$mass[new.cluster]/area2
      if (irrPolygons[[2]]==irrPolygons[[3]])
      {
        print(sprintf("Doubling Irregular Polygons list' capacity from %d to %d...",irrPolygons[[2]],irrPolygons[[2]]*2))
        irrPolygons < double.irrPolygon(irrPolygons)
      }


      if(clusters[[2]]==clusters[[3]])
      {
        print(sprintf("Doubling clusters capacity from %d to %d...",clusters[[2]],clusters[[2]]*2))
       m< matrix(0,clusters[[2]]*2,length(names(clusters[[1]])))
       m< as.data.frame(m)
        names(m) < names(clusters[[1]])
        for (i in c(1:length(names(clusters[[1]]))))
         m[,i] < c(clusters[[1]][,i],clusters[[1]][,i])
        clusters[[3]] < clusters[[2]]
      }


      #recording operations
      operations[[2]] < operations[[2]]+1
      operations[[1]][operations[[2]],] < c(cluster_to_divide,cluster_to_divide,new.cluster,1)
      if (operations[[2]]==operations[[3]])
        operations < double.operations(operations)

      mytree[[1]] < insertNode(mytree[[1]],IDleaf,"condition",opt[[1]],opt[[4]],opt[[2]],mytree[[2]]+1,mytree[[2]]+2)
      mytree[[2]] < mytree[[2]]+2

      if (mytree[[2]] > mytree[[3]])
        mytree < expanding.tree(mytree)


      mytree[[1]] < insertNode(mytree[[1]],mytree[[2]] 1,"cluster",NA,NA,cluster_to_divide,NA,NA)
      mytree[[1]] < insertNode(mytree[[1]],mytree[[2]],"cluster",NA,NA,new.cluster,NA,NA)

    }
```

```
    }

    npoints[np,] < c(length(total.clusters),count)
    npoints2 < npoints[1:(np+1),]
    str(npoints2)
    npoints2 < as.data.frame(npoints2)
    str(npoints2)
    names(npoints2) < c("Memory","Reality")
    write.csv2(npoints2,"point_ratio.csv")
    np < np+1

            #check merging criterion
    idxsel < which(clusters[[1]]$interest[1:clusters[[2]]]==0 & (clusters[[1]]$mass[1:clusters[[2]]]/total.mass)>(clusters[[1]]$initialMassRatio[1:clusters[[2]]]*merge.size.ratio))

    while (length(idxsel)>0)
    {
     idx < idxsel[1]
     changes < changes+1

     #get cluster to merge
     cluster_to_merge< clusters[[1]]$cluster[idx]
     print("merging cluster...")
     print(sprintf("Clustering to merge: %d",cluster_to_merge))

     idxclusteroriginal < which(operations[[1]]$new1[1:operations[[2]]]==cluster_to_merge | operations[[1]]$new2[1:operations[[2]]]==cluster_to_merge)
     idxclusteroriginal < max(idxclusteroriginal)

     #get clusters to merge
     original < operations[[1]]$original[idxclusteroriginal]
     new1 < operations[[1]]$new1[idxclusteroriginal]
     new2 < operations[[1]]$new2[idxclusteroriginal]

     if (original!=new1)
      new < new1
     else
      new < new2

     operations[[1]] < operations[[1]][ idxclusteroriginal ,]
     operations[[2]] < operations[[2]] 1
     operations[[3]] < operations[[3]] 1

     idxnode1 < which(mytree[[1]]$node_type[1:mytree[[2]]]=="cluster" & mytree[[1]]$value[1:mytree[[2]]]==original)
     idxnode2 < which(mytree[[1]]$node_type[1:mytree[[2]]]=="cluster" & mytree[[1]]$value[1:mytree[[2]]]==new)
     idxnode1 < mytree[[1]]$IDnode[idxnode1]
     idxnode2 < mytree[[1]]$IDnode[idxnode2]

```

```
    idxnode < which((mytree[[1]]$left==idxnode1 & mytree[[1]]$right==idxnode2) | (mytree[[1]]$left==idxnode2 & mytree[[1]]$right==idxnode1))[1]
    idxnode < mytree[[1]]$IDnode[idxnode]

    divise_type < mytree[[1]]$condition_type[which(mytree[[1]]$IDnode==idxnode)]
    latlon_value < mytree[[1]]$value[which(mytree[[1]]$IDnode==idxnode)]

    cluster_to_remove < max(original,new)
    cluster_to_maintain < min(original,new)

    mytree[[1]] < insertNode(mytree[[1]], idxnode,"cluster",NA,NA, cluster_to_maintain,NA,NA)

    cl1 < which(clusters[[1]]$cluster==cluster_to_maintain)
    cl2 < which(clusters[[1]]$cluster==cluster_to_remove)

    new.poly < merge.clusters(c(clusters[[1]][cl1,c(3:6)]),c(clusters[[1]][cl2,c(3:6)]),divise_type,latlon_value)
    mass < clusters[[1]]$mass[cl1]+clusters[[1]]$mass[cl2]

    new.meanlat < ((clusters[[1]]$meanlat[cl1]*clusters[[1]]$mass[cl1])+(clusters[[1]]$meanlat[cl2]*clusters[[1]]$mass[cl2]))/(clusters[[1]]$mass[cl1]+clusters[[1]]$mass[cl2])
    new.meanlon < ((clusters[[1]]$meanlon[cl1]*clusters[[1]]$mass[cl1])+(clusters[[1]]$meanlon[cl2]*clusters[[1]]$mass[cl2]))/(clusters[[1]]$mass[cl1]+clusters[[1]]$mass[cl2])

    clusters[[1]][cl1,] < c(cluster_to_maintain,mass,new.poly[1],new.poly[2],new.poly[3],new.poly[4],0,0,0,1,new.meanlat,new.meanlon,mass/total.mass)

    obj < updateIPoly(clusters,cluster_to_maintain,irrPolygons,getIrregularPolygon(new.poly[1],new.poly[2],new.poly[3],new.poly[4],obst))
    irrPolygons < obj[[2]]
    clusters < obj[[1]]

    #updating density...
    area2 < calcAreaIrr(cl1,clusters[[1]],irrPolygons[[1]])
    clusters[[1]]$density[cl1] < clusters[[1]]$mass[cl1]/area2

    idx < which(clusters[[1]]$cluster>=cluster_to_remove)

    clusters[[1]]$cluster[idx] < clusters[[1]]$cluster[idx] 1
    clusters[[1]] < clusters[[1]][ cl2,]
    clusters[[2]] < clusters[[2]] 1
    clusters[[3]] < clusters[[3]] 1

```

```
      idx < which(mytree[[1]]$node_type==" cluster" & mytree[[1]]$
          value>=cluster_to_remove)
      mytree[[1]]$value[idx] < mytree[[1]]$value[idx] 1

      idx < which(operations[[1]]$original>=cluster_to_remove)
      operations[[1]]$original[idx] < operations[[1]]$original[
          idx] 1

      idx < which(operations[[1]]$new1>=cluster_to_remove)
      operations[[1]]$new1[idx] < operations[[1]]$new1[idx] 1

      idx < which(operations[[1]]$new2>=cluster_to_remove)
      operations[[1]]$new2[idx] < operations[[1]]$new2[idx] 1

      idxsel < which(clusters[[1]]$interest[1:clusters[[2]]]==0 &
          (clusters[[1]]$mass[1:clusters[[2]]]/total.mass)>(
          clusters[[1]]$initialMassRatio[1:clusters[[2]]]*1.5))
    }

    if (changes>0)
    {
                total.clusters < run_tree(my.ds, mytree, TRUE)
    }

    idx < which(clusters[[1]]$interest[1:clusters[[2]]] >0 & (
        clusters[[1]]$mass[1:clusters[[2]]])<(interest.ratio*
        total.mass))
    if (length(idx)>0)
    {
      clusters[[1]]$interest[idx] < 0
    }
   }

   if ((count%%plot.step)==0)
   {

    if(plotting)
      spatial_clustering("mass clustering", clusters[[2]], my.ds,
         NULL, zoom=12, total.clusters, list(clusters[[1]][,1:10],
         color_pallette), FALSE, FALSE, TRUE, TRUE, TRUE, count+
         initialK, c("pdf", "black"), map.center, irrPolygons)
   }
  }
}
```

Bibliography

M. Abkowitz and I. Engelstein. Methods for maintaining transit service regularity. Technical report, 1984.

M. Abkowitz and J. Tozzi. Research contributions to managing transit service reliability. *Journal of advanced transportation*, 21(1):47–65, 1987.

B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '02, pages 633–634, Philadelphia, PA, USA, 2002. Society for Industrial and Applied Mathematics.

G. Ball and D. Hall. Isodata, a novel method of data analysis and pattern classification. Technical report, DTIC Document, 1965.

A. Baptista, E. Bouillet, and P. Pompey. Towards an uncertainty aware short-term travel time prediction using gps bus data: Case study in dublin. In *15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1620–1625. IEEE, 2012.

B. Barabino, M. Francesco, and S. Mozzoni. Regularity diagnosis by automatic vehicle location raw data. *Public Transport*, 4(3):187–208, 2013.

D. Barbará. Requirements for clustering data streams. *SIGKDD Explor. Newsl.*, 3(2):23–27, January 2002.

J. Barceló, L. Montero, M. Bullejos, O. Serch, and C. Carmona. A kalman filter approach for exploiting bluetooth traffic data when estimating time-dependent od matrices. *Journal of Intelligent Transportation Systems*, 17(2):123–141, 2013.

Beijing City Lab. T-drive taxi trajectories, January 2015.

G. Bellei and K. Gkoumas. Transit vehicles' headway distribution and service irregularity. *Public Transport*, 2(4):269–289, 2010.

M. Berkow, J. Chee, R. Bertini, and C. Monsere. Transit performance measurement and arterial travel time estimation using archived avl data. In *ITE District 6 Annual Meeting*, 2007.

R. Bertini and A. El-Geneidy. Generating transit performance measures with archived data. *Transportation Research Record: Journal of the Transportation Research Board*, 1841(1):109–119, 2003.

R. Bertini and A. El-Geneidy. Modeling transit trip time using archived bus dispatch system data. *Journal of transportation engineering*, 130(1):56–67, 2003.

Y. Bin, Y. Zhongzhen, and Y. Baozhen. Bus arrival time prediction using support vector machines. *Journal of Intelligent Transportation Systems*, 10(4):151–158, 2006.

L. Birge and Y. Rozenholc. How many bins should be put in a regular histogram. *ESAIM: Probability and Statistics*, 10:24–45, 10 2006.

C. Bishop et al. *Neural networks for pattern recognition.* Clarendon press Oxford, 1995.

G. Box, G. Jenkins, and G. Reinsel. *Time series analysis.* Holden-day San Francisco, 1976.

G. Box. Non-normality and tests on variances. *Biometrika*, 40(3/4):318–335, 1953.

L. Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

A. Broder, M. Charikar, A. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630 – 659, 2000.

E. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. Woodward. A classification of hyper-heuristic approaches. In *Handbook of Metaheuristics*, volume 146, pages 449–468. Springer US, 2010.

F. Calabrese, M. Colonna, P. Lovisolo, D. Parata, and C. Ratti. Real-time urban monitoring using cell phones: A case study in rome. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):141–151, 2011.

O. Cappé, S. Godsill, and E. Moulines. An overview of existing methods and recent advances in sequential monte carlo. *Proceedings of the IEEE*, 95(5):899–924, 2007.

P. Carnaghi. Decision support systems in transportation: the case of surface urban public transport. Master's thesis, 2014.

U. Carrascal. A review of travel time estimation and forecasting for advanced traveler information systems. Master's thesis, Universidad del Pais Vasco, 2012.

P. Castro, D. Zhang, C. Chen, S. Li, and G. Pan. From taxi gps traces to social and community dynamics: A survey. *ACM Comput. Surv.*, 46:17:1–17:34, 2013.

F. Cathey and D. Dailey. A prescription for transit arrival/departure prediction using automatic vehicle location data. *Transportation Research Part C: Emerging Technologies*, 11(3):241–264, 2003.

O. Cats, W. Burghout, T. Toledo, and H. Koutsopoulos. Evaluation of real-time holding strategies for improved bus service reliability. In *13th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 718–723, 2010.

O. Cats, An. Larijani, W. Ólafsdóttir, A.and Burghout, I. Andréasson, and H. Koutsopoulos. Bus-holding control strategies. *Transportation Research Record: Journal of the Transportation Research Board*, 2274(1):100–108, 2012.

O. Cats, F. Rufi, and H. Koutsopoulos. Optimizing the number and location of time point stops. *Public Transport*, 6(3):215–235, 2014.

O. Cats. Regularity-driven bus operation: Principles, implementation and business models. *Transport Policy*, 36:223–230, 2014.

A. Ceder and P. Marguier. Passenger waiting time at transit stops. *Traffic engineering & control*, 26(6):327–329, 1985.

A. Ceder. Urban transit scheduling: framework, review and examples. *Journal of Urban Planning and Development*, 128(4):225–244, 2002.

A. Ceder. *Public transit planning and operation: theory, modeling and practice*. Elsevier, Butterworth-Heinemann, 2007.

S. Chalup. Incremental learning in biological and machine learning systems. *International Journal of Neural Systems*, 12(06):447–465, 2002.

L. Cham. *Understanding bus service reliability: a practical framework using AVL/APC data*. PhD thesis, Massachusetts Institute of Technology, 2006.

H. Chang, D. Park, S. Lee, H. Lee, and S. Baek. Dynamic multi-interval bus travel time prediction using bus transit data. *Transportmetrica*, 6(1):19–38, 2010.

H. Chang, Y. Tai, and J. Hsu. Context-aware taxi demand hotspots prediction. *International Journal of Business Intelligence and Data Mining*, 5(1):3–18, 2010.

R. Chapman and J. Michel. Modelling the tendency of buses to form pairs. *Transportation Science*, 12(2):165–175, 1978.

W. Chen and Z. Chen. A simulation model for transit service unreliability prevention based on avl-apc data. In *International Conference on Measuring Technology and Mechatronics Automation*, volume 2, pages 184–188. IEEE, 2009.

M. Chen, X. Liu, J. Xia, and S. Chien. A dynamic bus-arrival time prediction model based on apc data. *Computer-Aided Civil and Infrastructure Engineering*, 19(5):364–376, 2004.

B. Chen, L. Chen, Y. Lin, and R. Ramakrishnan. Prediction cubes. In *Proceedings of the 31st international conference on Very large data bases*, pages 982–993. VLDB Endowment, 2005.

G. Chen, X. Yang, J. An, and D. Zhang. Bus-arrival-time prediction models: Link-based and section-based. *Journal of Transportation Engineering*, 138(1):60–66, 2011.

C. Chen, D. Zhang, P. Castro, N. Li, L. Sun, and S. Li. Real-time detection of anomalous taxi trajectories from gps traces. In *Mobile and Ubiquitous Systems: Computing, Networking, and Services*, pages 63–74. Springer, 2012.

S. Cheng and T. Nguyen. Taxisim: A multiagent simulation platform for evaluating taxi fleet operations. In *Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 02*, pages 14–21. IEEE Computer Society, 2011.

H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

S. Chien, Y. Ding, and C. Wei. Dynamic bus arrival time prediction with artificial neural networks. *Journal of Transportation Engineering*, 128(5):429–438, 2002.

S. Chu, E. Keogh, D. Hart, M. Pazzani, et al. Iterative deepening dynamic time warping for time series. In *Second SIAM International Conference on Data Mining (SDM-02)*, 2002.

E. Chung and A. Shalaby. Expected time of arrival model for school bus transit using real-time global positioning system-based automatic vehicle location data. *Journal of Intelligent Transportation Systems*, 11(4):157–167, 2007.

W. Cleveland. Lowess: A program for smoothing scatterplots by robust locally weighted regression. *The American Statistician*, 35(1):54–54, 1981.

C. Clotfelter. The private life of public economics. *Southern Economic Journal*, pages 579–596, 1993.

W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning, Lake Tahoe, California*, 1995.

J. Contreras, R. Espinola, F. Nogales, and A. Conejo. Arima models to predict next-day electricity prices. *IEEE Transactions on Power Systems*, 18(3):1014–1020, 2003.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

J. Cryer and K. Chan. *Time Series Analysis with Applications in R.* Springer, USA, 2008.

C. Daganzo and J. Pilachowski. Reducing bunching with bus-to-bus cooperation. *Transportation Research Part B: Methodological*, 45(1):267–277, 2011.

C. Daganzo. A headway-based approach to eliminate bus bunching. *Transportation Research Part B*, 43(10):913–921, 2009.

M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining stream statistics over sliding windows. *SIAM Journal on Computing*, 31(6):1794–1813, 2002.

F. Delgado, J. Muñoz, R. Giesen, and A. Cipriano. Real-time control of buses in a transit corridor based on vehicle holding and boarding limits. *Transportation Research Record: Journal of the Transportation Research Board*, 2090(1):59–67, 2009.

F. Delgado, J. Munoz, and R. Giesen. How much can holding and/or limiting boarding improve transit performance? *Transportation Research Part B: Methodological*, 46(9):1202–1217, 2012.

Z. Deng and M. Ji. Spatiotemporal structure of taxi services in shanghai: Using exploratory spatial data analysis. In *19th International Conference on Geoinformatics*, pages 1–5. IEEE, 2011.

M. Dessouky, R. Hall, A. Nowroozi, and K. Mourikas. Bus dispatching at timed transfer transit stations using bus tracking technology. *Transportation Research Part C: Emerging Technologies*, 7(4):187–208, 1999.

M. Dessouky, R. Hall, L. Zhang, and A. Singh. Real-time control of buses for schedule coordination at a terminal. *Transportation Research Part A: Policy and Practice*, 37(2):145–164, 2003.

T. Dietterich. Machine-learning research. *AI magazine*, 18(4):97, 1997.

M. Dixon and L. Rilett. Real-time od estimation using automatic vehicle identification and traffic count data. *Computer-Aided Civil and Infrastructure Engineering*, 17(1):7–21, 2002.

T. Do and D. Gatica-Perez. Where and what: Using smartphones to predict next locations and applications in daily life. *Pervasive and Mobile Computing*, 12(0):79 – 91, 2014.

P. Domingos and M. Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine learning*, 29(2-3):103–130, 1997.

J. Dong, L. Zou, and Y. Zhang. Mixed model for prediction of bus arrival times. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 2918–2923. IEEE, 2013.

M. Dorigo and L. Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik. Support vector regression machines. *Advances in neural information processing systems*, pages 155–161, 1997.

Dublinked. Dublin bus gtfs data, January 2015.

X. Eberlein, N. Wilson, and D. Bernstein. Modeling real-time control strategies in public transit operations. In *Computer-aided Transit Scheduling*, pages 325–346. Springer, 1999.

X. Eberlein, N. Wilson, and Bernstein. The holding problem with real-time information available. *Transportation Science*, 35(1):1–18, 2001.

A. El-Geneidy and J. Surprenant-Legault. Limited-stop bus service: an evaluation of an implementation strategy. *Public Transport*, 2(4):291–306, 2010.

A. El-Geneidy, J. Horning, and K. Krizek. Analyzing transit service reliability using detailed data from automatic vehicular locator systems. *Journal of Advanced Transportation*, 45(1):66–79, 2011.

M. Ester, H. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

European Parliament. Directive 2010/40/eu of the european parliament and of the council, July 2010.

G. Fattouche. *Improving high-frequency bus service reliability through better scheduling*. PhD thesis, Massachusetts Institute of Technology, 2007.

M. Ferreira, H. Conceição, R. Fernandes, and O. Tonguz. Stereoscopic aerial photography: an alternative to model-based urban mobility approaches. In *Proceedings of the sixth ACM international workshop on VehiculAr InterNETworking*, pages 53–62. ACM, 2009.

M. Ferreira, R. Fernandes, H.o Conceição, W. Viriyasitavat, and O. Tonguz. Self-organized traffic control. In *Proceedings of the seventh ACM international workshop on VehiculAr InterNETworking*, pages 85–90. ACM, 2010.

C Ferreira, J Gama, L Matias, A Botterud, and J Wang. A survey on wind power ramp forecasting. Technical report, Argonne National Laboratory (ANL), 2011.

M. Ferreira, R. Fernandes, H. Conceição, P. Gomes, P. dOrey, L. Moreira-Matias, J. Gama, F. Lima, and L. Damas. *Vehicular Sensing: Emergence of a Massive Urban Scanner*, volume 102 of *LNCS: Social Informatics and Telecommunications Engineering*, pages 1–14. Springer, 2012.

J. Friedman and Werner Stuetzle. Projection pursuit regression. *Journal of the American statistical Association*, 76(376):817–823, 1981.

L. Fu and X. Yang. Design and implementation of bus-holding control strategies with real-time information. *Transportation Research Record: Journal of the Transportation Research Board*, 1791(1):6–12, 2002.

P. Furth, B. Hemily, T. Muller, and J. Strathman. *Uses of archived AVL-APC data to improve transit performance and management: Review and potential*. Transportation Research Board, 2003.

J. Gama and C. Pinto. Discretization from data streams: applications to histograms and data mining. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, pages 662–667. ACM, 2006.

Joao Gama, Raquel Sebastiao, and Pedro Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013.

J. Gama. *Knowledge discovery from data streams.* Chapman and Hall/CRC, 2010.

Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani. An energy-efficient mobile recommender system. In *Proceedings of the 16th ACM SIGKDD international Conference on Knowledge Discovery and Data Mining*, pages 899–908. ACM, 2010.

Y. Ge, H. Xiong, C. Liu, and Z. Zhou. A taxi driving fraud detection system. In *11th International Conference on Data Mining (ICDM)*, pages 181–190. IEEE, 2011.

G. Gentile, S. Nguyen, and S. Pallottino. Route choice on transit networks with online information at stops. *Transportation Science*, 39(3):289–297, 2005.

B. Ghosh and P. Sen. *Handbook of sequential analysis*, volume 118. CRC Press, 1991.

P. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. In *Proceedings of the International Conference on Very Large Data Bases*, pages 466–475. IEEE, 1997.

Z. Gurmu. *A Dynamic Prediction of Travel Time for Transit Vehicles in Brazil Using GPS Data.* PhD thesis, University of Twente, 2010.

Y. Hadas and A. Ceder. Optimal coordination of public-transit vehicles using operational tactics examined by simulation. *Transportation Research Part C: Emerging Technologies*, 18(6):879–895, 2010.

Y. Hadas and M. Shnaiderman. Public-transit frequency setting using minimum-cost approach with stochastic demand and travel time. *Transportation Research Part B: Methodological*, 46(8):1068–1084, 2012.

M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

R. Hariharan and K. Toyama. Project lachesis: parsing and modeling location histories. In *Geographic Information Science*, pages 106–124. Springer, 2004.

M. Hickman. An analytic stochastic model for the transit vehicle holding problem. *Transportation Science*, 35(3):215–237, 2001.

W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004.

G. Hong-Cheng, Y. Xin, and W. Qing. Investigating the effect of travel time variability on drivers' route choice decisions in shanghai, china. *Transportation Planning and Technology*, 33(8):657–669, 2010.

M. Hoque, X. Hong, and B. Dixon. Analysis of mobility patterns for urban taxi cabs. In *2012 International Conference on Computing, Networking and Communications (ICNC)*, pages 756–760. IEEE, 2012.

A. Horbury. Using non-real-time automatic vehicle location data to improve bus services. *Transportation Research Part B: Methodological*, 33(8):559–579, 1999.

N. Hounsell and F. McLeod. Automatic vehicle location: Implementation, application, and benefits in the united kingdom. *Transportation Research Record: Journal of the Transportation Research Board*, 1618(1):155–162, 1998.

N. Hounsell, B. Shrestha, and A. Wong. Data management and applications in a world-leading bus fleet. *Transportation Research. Part C, Emerging Technologies*, 22:76–87, 2012.

H. Hu, Z. Wu, B. Mao, Y. Zhuang, J. Cao, and J. Pan. Pick-up tree based route recommendation from taxi trajectories. In *Web-Age Information Management*, pages 471–483. Springer, 2012.

G. Huang, Q. Zhu, and C. Siew. Real-time learning capability of neural networks. *IEEE Transactions on Neural Networks*, 17(4):863 – 878, july 2006.

A. Ihler, J. Hutchins, and P. Smyth. Adaptive event detection with time-varying poisson processes. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 207–216. ACM, 2006.

A. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31(8):656–666, 2010.

E. Jaynes. *Probability theory: The logic of science.* Cambridge university press, 2003.

R. Jeong and L. Rilett. Prediction model of bus arrival time for real-time applications. *Transportation Research Record: Journal of the Transportation Research Board*, 1927(1):195–204, 2005.

A. Jorge, J. Mendes-Moreira, J. Freire de Sousa, C. Soares, and P. Azevedo. Finding interesting contexts for explaining deviations in bus trip duration using distribution rules. In *IDA*, pages 139–149, 2012.

R. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

A. Khosravi, E. Mazloumi, S. Nahavandi, and D. Creighton. Prediction intervals to account for uncertainties in travel time prediction. *Transportation Research Part C: Emerging Technologies*, 19(2):1364–1376, 2011.

A. Khosravi, E. Mazloumi, S. Nahavandi, D. Creighton, and J. Van Lint. Prediction intervals to account for uncertainties in travel time prediction. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):537–547, 2011.

L. Kieu, A. Bhaskar, and E. Chung. Benefits and issues for bus travel time estimation and prediction. In *Australasian Transport Research Forum 2012*, Perth, WA, September 2012.

G. Klunder, P. Baas, and F. op de Beek. A long-term travel time prediction algorithm using historical data. In *Proceedings of the 14th World Congress on Intelligent Transport Systems*, 2007.

H. Koutsopoulos and Z. Wang. Simulation of urban rail operations: Application framework. *Transportation Research Record: Journal of the Transportation Research Board*, 2006(1):84–91, 2007.

J. Lee, G. Park, H. Kim, Y. Yang, P. Kim, and S. Kim. *A Telematics Service System Based on the Linux Cluster*, volume 4490 of *LNCS*, pages 660–667. Springer Berlin / Heidelberg, 2007.

J. Lee, I. Shin, and G. Park. Analysis of the passenger pick-up pattern for taxi location recommendation. In *Fourth International Conference on Networked Computing and Advanced Information Management (NCM'08)*, volume 1, pages 199–204. IEEE, 2008.

W. Lee, W. Si, L. Chen, and M. Chen. Http: a new framework for bus travel time prediction based on historical trajectories. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 279–288. ACM, 2012.

L. Lesley. The role of the timetable in maintaining bus service reliability. In *Proceedings of operating public transport symposium*, 1975.

Q. Li, Z. Zeng, B. Yang, and T. Zhang. Hierarchical route planning based on taxi gps-trajectories. In *17th International Conference on Geoinformatics*, pages 1–5. IEEE, 2009.

B. Li, D. Zhang, L. Sun, C. Chen, S. Li, G. Qi, and Q. Yang. Hunting or waiting? discovering passenger-finding strategies from a large-scale real-world taxi dataset. In *2011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 63–68, March 2011.

J. Li, T. Muller, H. van Zuylen, X. Chen, and W. Wang. Improving the reliability of transit service: Stochastic modeling of holding strategies. In *Transportation Research Board 90th Annual Meeting*, 2011.

X. Li, G. Pan, Z.ohui Wu, G. Qi, S. Li, D. Zhang, W. Zhang, and Z. Wang. Prediction of urban human mobility using large-scale taxi traces and its applications. *Frontiers of Computer Science in China*, 6(1):111–121, 2012.

Z. Liao. Taxi dispatching via global positioning systems. *IEEE Transactions on Engineering Management*, 48(3):342–347, 2001.

Z. Liao. Real-time taxi dispatching using global positioning systems. *Communications of the ACM*, 46(5):81–83, 2003.

W. Lin and R. Bertini. Modeling schedule recovery processes in transit operations for bus arrival time prediction. *Journal of Advanced Transportation*, 38(3):347–365, 2004.

J. Lin and M. Ruan. Probability-based bus headway regularity measure. *IET intelligent transport systems*, 3(4):400–408, 2009.

J. Lin, P. Wang, and D. Barnum. A quality control framework for bus schedule reliability. *Transportation Research Part E: Logistics and Transportation Review*, 44(6):1086–1098, 2008.

N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212 – 261, 1994.

N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2(4):285–318, 1988.

L. Liu, C. Andris, A. Biderman, and C. Ratti. Uncovering taxi drivers mobility intelligence through his trace. *IEEE Pervasive Computing*, 160:1–17, 2009.

L. Liu, C. Andris, A. Bidderman, and C. Ratti. Revealing taxi drivers mobility intelligence through his trace. *Movement-Aware Applications for Sustainable Mobility: Technologies and Approaches*, pages 105–120, 2010.

L. Liu, C. Andris, and C. Ratti. Uncovering cabdrivers behavior patterns from their digital traces. *Computers, Environment and Urban Systems*, 34(6):541–548, 2010.

T. Liu, J. Ma, W. Guan, Y. Song, and H. Niu. Bus arrival time prediction based on the k-nearest neighbor method. In *Computational Sciences and Optimization (CSO), 2012 Fifth International Joint Conference on*, pages 480–483. IEEE, 2012.

Z. Liu, Y. Yan, X. Qu, and Y. Zhang. Bus stop-skipping scheme with random travel time. *Transportation Research Part C: Emerging Technologies*, 35:46–56, 2013.

S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, 1982.

J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, page 14. California, USA, 1967.

M. Mandelzys and B. Hellinga. Identifying causes of performance issues in bus schedule adherence with automatic vehicle location and passenger count data. *Transportation Research Record: Journal of the Transportation Research Board*, 2143(1):9–15, 2010.

A. Markov. The theory of algorithms. *Trudy Matematicheskogo Instituta im. VA Steklova*, 42:3–375, 1954.

L. Matias, J. Gama, J. Mendes-Moreira, and J. Freire de Sousa. Validation of both number and coverage of bus schedules using avl data. In *13th IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 131–136, 2010.

M. Mayuri and M. Rajesh. Gps trajectories based system: T-finder. *IOSR Journal of Computer Engineering (IOSR-JCE)*, pages 20–24, 2013.

E. Mazloumi, G. Currie, and G. Rose. Using gps data to gain insight into public transport travel time variability. *Journal of Transportation Engineering*, 136(7):623–631, 2010.

E. Mazloumi, G. Rose, G. Currie, and S. Moridpour. Prediction intervals to account for uncertainties in neural network predictions: Methodology and application in bus travel time prediction. *Engineering Applications of Artificial Intelligence*, 24(3):534–542, 2011.

E. Mazloumi, M. Mesbah, A. Ceder, S. Moridpour, and G. Currie. Efficient transit schedule design of timing points: A comparison of ant colony and genetic algorithms. *Transportation Research Part B: Methodological*, 46(1):217–234, 2012.

J. McClelland, D. Rumelhart, PDP Research Group, et al. Parallel distributed processing. *Explorations in the microstructure of cognition*, 2, 1986.

M. Melanie. An introduction to genetic algorithms. *Cambridge, Massachusetts London, England, Fifth printing*, 3, 1999.

J. Mendes-Moreira and J.F. Sousa. Evaluating changes in the operational planning of public transportation. In *Computer-based Modelling and Optimization in Transportation*, volume 262 of *Advances in Intelligent Systems and Computing*, pages 57–68. Springer International Publishing, 2014.

J. Mendes-Moreira, A. Jorge, J. de Sousa, and C. Soares. Comparing state-of-the-art regression methods for long term travel time prediction. *Intelligent Data Analysis*, 16(3):427–449, 2012.

Joao Mendes-Moreira, Luis Moreira-Matias, Joao Gama, and Jorge Freire de Sousa. Validating the coverage of bus schedules: A machine learning approach. *Information Sciences*, 293(0):299 – 313, 2015.

J. Mendes-Moreira. *Travel Time Prediction for the Planning of Mass Transit Companies: a Machine Learning Approach.* PhD thesis, University of Porto, 2008.

W. Min and L. Wynter. Real-time road traffic prediction with spatio-temporal correlations. *Transportation Research Part C: Emerging Technologies*, 19(4):606–616, 2011.

M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of machine learning.* MIT press, 2012.

S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, 52(1):91–118, 2003.

L. Moreira-Matias, R. Fernandes, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas. An online recommendation system for the taxi stand choice problem (poster). In *IEEE Vehicular Networking Conference (VNC)*, pages 173–180, 2012.

L. Moreira-Matias, C. Ferreira, J. Gama, J. Mendes-Moreira, and J. de Sousa. Bus bunching detection by mining sequences of headway deviations. In *Advances in Data Mining. Applications and Theoretical Aspects*, volume 7377 of *LNCS*, pages 77–91. Springer., 2012.

L. Moreira-Matias, C. Ferreira, J. Gama, J. Mendes-Moreira, and J.F. de Sousa. Bus bunching detection: A sequence mining approach. In *Workshop on Ubiquitous Data Mining*, page 13, 2012.

L. Moreira-Matias, J. Gama, M. Ferreira, and L. Damas. A predictive model for the passenger demand on a taxi network. In *15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1014 –1019, sept. 2012.

L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas. Online predictive model for taxi services. In *Advances in Intelligent Data Analysis XI*, volume 7619 of *LNCS*, pages 230–240. Springer Berlin Heidelberg, 2012.

L. Moreira-Matias, J. Mendes-Moreira, J. Gama, and P. Brazdil. Text categorization using an ensemble classifier based on a mean co-association matrix. In *Machine Learning and Data Mining in Pattern Recognition*, volume 7376 of *LNCS*, pages 525–539. Springer Berlin / Heidelberg, 2012.

L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas. On predicting the taxi-passenger demand: A real-time approach. In *Progress in Artificial Intelligence*, volume 8154 of *LNCS*, pages 54–65. Springer, 2013.

L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas. Predicting taxi-passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1393–1402, 2013.

Luis Moreira-Matias, Ricardo Fernandes, Joao Gama, Michel Ferreira, Joao Mendes-Moreira, and Luis Damas. On recommending urban hotspots to find our next passenger. In *Ubiquitous Data Mining workshop held in conjunction with the 23rd. International Joint Conference on Artificial Intelligence (UDM/IJCAI)*, page 17, 2013.

L. Moreira-Matias, J. Gama, J. Mendes-Moreira, and J. Freire de Sousa. An incremental probabilistic model to predict bus bunching in real-time. In *Advances in Intelligent Data Analysis XIII*, volume 8819 of *LNCS*, pages 227–238. Springer International Publishing, 2014.

Luis Moreira-Matias, Joao Mendes-Moreira, Michel Ferreira, Joao Gama, and Luis Damas. An online learning framework for predicting the taxi stand's profitability. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 2009–2014, Oct 2014.

Luis Moreira-Matias, Joao Mendes-Moreira, Joao Gama, and Michel Ferreira. On improving operational planning and control in public transportation networks using streaming data: A machine learning approach. In *Proceedings of the European Conference of Machine Learning and Principles of Knowledge Discovery on Databases (ECML/PKDD, Phd Spotlight Session)*, page 41, 2014.

Luis Moreira-Matias, Rafael Nunes, Michel Ferreira, Joao Mendes-Moreira, and Joao Gama. On predicting a call centers workload: A discretization-based approach. In *Foundations of Intelligent Systems*, volume 8502 of *LNCS*, pages 548–553. Springer International Publishing, 2014.

L. Moreira-Matias, J. Mendes-Moreira, J. Freire de Sousa, and J. Gama. On improving mass transit operations by using avl-based systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, In Press, 2015.

M. Munizaga and C. Palma. Estimation of a disaggregate multimodal public transport origin–destination matrix from passive smartcard data from santiago, chile. *Transportation Research Part C: Emerging Technologies*, 24:9–18, 2012.

E. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

Y. Nakanishi. Bus performance indicators: On-time performance and service regularity. *Transportation Research Record: Journal of the Transportation Research Board*, 1571:1–13, 1997.

G. Newell and R. Potts. Maintaining a bus schedule. In *2nd Australian Road Research Board*, volume 2, pages 388–393, 1964.

R. Nunes, L. Moreira-Matias, and M. Ferreira. Using exit time predictions to optimize self automated parking lots. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 302 –307, oct. 2012.

Rafael Nunes, Luis Moreira-Matias, and Michel Ferreira. Using exit time predictions to optimize self automated parking lots. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 302–307. IEEE, 2014.

N. Oza. *Online Ensemble Learning*. PhD thesis, University of California, 2001.

E. Page. Continuous inspection schemes. *Biometrika*, 41(1/2):100–115, 1954.

D. Park and L. Rilett. Forecasting freeway link travel times with a multilayer feedforward neural network. *Computer-Aided Civil and Infrastructure Engineering*, 14(5):357–367, 1999.

J. Patnaik, S. Chien, and A. Bladikas. Estimation of bus arrival times using apc data. *Journal of Public Transportation*, 7(1):1–20, 2004.

J. Patnaik, S. Chien, and A. Bladikas. Using data mining techniques on apc data to develop effective bus scheduling. *Journal of Systemics, Cybernetics and Informatics*, 4(1):86–90, 2006.

G. Peek and M. van Hagen. Creating synergy in and around stations: Three strategies for adding value. In *Transportation Research Board*, volume 1793, pages 1–6, 2002.

S. Phithakkitnukoon, M. Veloso, C. Bento, A. Biderman, and C. Ratti. Taxi-aware map: identifying and predicting vacant taxis in the city. *Ambient Intelligence*, 6439:86–95, 2010.

W. Powell and Y. Sheffi. A probabilistic model of bus route performance. *Transportation Science*, 17(4):376–404, 1983.

J. Powell, Y. Huang, F. Bastani, and M. Ji. Towards reducing taxicab cruising time using spatio-temporal profitability maps. In *Advances in Spatial and Temporal Databases*, pages 242–260. Springer, 2011.

G. Qi, X. Li, S. Li, G. Pan, Z. Wang, and D. Zhang. Measuring social functions of city regions from large-scale taxi behaviors. In *IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, pages 384–388. IEEE, 2011.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012.

R. Rajbhandari. *Bus arrival time prediction using stochastic time series and markov chains*. PhD thesis, New Jersey Institute of Technology, 2005.

C. Robusto. The cosine-haversine formula. *The American Mathematical Monthly*, 64(1):38–40, 1957.

P. Rodrigues and J. Gama. A system for analysis and prediction of electricity-load streams. *Intelligent Data Analysis*, 13(3):477–496, 2009.

F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

D. Sáez, C. Cortés, F. Milla, A. Núñez, A. Tirachini, and M. Riquelme. Hybrid predictive control strategy for a public transport system with uncertain demand. *Transportmetrica*, 8(1):61–86, 2012.

Schaller Consulting. *The New York City Taxicab Fact Book*. Schaller Consulting, 2006.

B. Schaller. Entry controls in taxi regulation: Implications of us and canadian experience for taxi regulation and deregulation. *Transport Policy*, 14(6):490–506, 2007.

R. Schapire. The strength of weak learnability. *Machine learning*, 5(2):197–227, 1990.

D. Schrank, B. Eisele, and T. Lomax. Ttis 2012 urban mobility report. Technical report, Texas A&M Transportation Institute, 2012.

A. Shalaby and A. Farhan. Bus travel time prediction model for dynamic operations control and passenger information systems. *Transportation Research Board*, 2003.

Shanghai Municipal Statistical Bureau. *Shanghai Statistical Yearbook 2008*. China statistics press, 2008.

M. Sinn, J. Yoon, F. Calabrese, and E. Bouillet. Predicting arrival times of buses using real-time gps measurements. In *2012 15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1227–1232. IEEE, 2012.

STCP - Sociedade de Transportes Colectivos do Porto, October 2013.

B. Sterman and J. Schofer. Factors affecting reliability of urban bus services. *Journal of Transportation Engineering*, 102(ASCE# 11930), 1976.

G. Stone. An analysis of the delta rule and the learning of statistical associations. *Parallel distributed processing: Explorations in the microstructure of cognition*, 1:444–459, 1986.

J. Strathman, T. Kimpel, and K. Dueker. Automated bus dispatching, operations control and service reliability. *Transportation Research Record*, 1666:28–36, 1999.

J. Strathman, T. Kimpel, K. Dueker, and Transportation Northwest. Bus transit operations control: review and an experiment involving tri-met's automated bus dispatching system. Technical report, Transportation Northwest, Department of Civil Engineering, University of Washington, 2000.

J. Strathman, T. Kimpel, S. Callas, and T. Northwest. Headway deviation effects on bus passenger loads: Analysis of tri-met's archived avl-apc data. Technical report, Citeseer, 2003.

J. Strathman. Automated bus dispatching, operations control and service reliability: analysis of tri-met baseline service date. Technical report, University of Washington, 1998.

J. Strathman. Tri-met's experience with automatic passenger counter and automatic vehicle location systems. *Center for Urban Studies, Portland State University*, 2002.

H. Sturges. The choice of a class interval. *Journal of the American Statistical Association*, 21(153):65–66, 1926.

T. Su and J. Dy. A deterministic method for initializing k-means clustering. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 784 – 786, 2004.

A. Sun and M. Hickman. The holding problem at multiple holding stations. In *Computer-aided systems in public transport*, pages 339–359. Springer, 2008.

D. Sun, H. Luo, L. Fu, W. Liu, X. Liao, and M. Zhao. Predicting bus arrival time on the basis of global positioning system data. *Transportation Research Record: Journal of the Transportation Research Board*, 2034(1):62–72, 2007.

W. Suwardo, M. Napiah, and I. Kamaruddin. Arima models for bus travel time prediction. *The Journal of the Institution of Engineers*, 71(2):49–58, 2010.

C. Tan, S. Park, H. Liu, Q. Xu, and P. Lau. Prediction of transit vehicle arrival time for signal priority control: algorithm and performance. *IEEE Transactions on Intelligent Transportation Systems*, 9(4):688–696, 2008.

N. Tatbul, U. Çetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker. Load shedding in a data stream manager. In *Proceedings of the 29th international conference on Very large data bases - Volume 29*, VLDB '03, pages 309–320. VLDB Endowment, 2003.

TCRP. *Transit Capacity and Quality of Service Manual*, volume 100. Transportation Research Board, 2003.

P. Tétreault and A. El-Geneidy. Estimating bus run times for new limited-stop service using archived avl and apc data. *Transportation Research Part A: Policy and Practice*, 44(6):390–402, 2010.

A. Theiss, D. Yen, and C. Ku. Global positioning systems: an analysis of applications, current development and future implementations. *Computer Standards & Interfaces*, 27(2):89 – 100, 2005.

D. Tiesyte and C. Jensen. Similarity-based prediction of travel times for vehicles traveling on known routes. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, page 14. ACM, 2008.

K. Ting and J. Wells. Multi-dimensional mass estimation and mass-based clustering. In *IEEE 10th International Conference on Data Mining (ICDM)*, pages 511–520, 2010.

Transportation Research Board. *Transit Capacity and Quality of Service Manual*, volume 100. Transportation Research Board, 2003.

H. Trevor, T. Robert, and J. Friedman. *The elements of statistical learning*, volume 1. Springer New York, 2001.

Tri-Met. Operations control plan for the tri-county metropolitan transportation district of oregon (tri-met). Technical report, Tri-County Metropolitan Transportation District of Oregon, 1991.

M. Trompet, X. Liu, and D. Graham. Development of key performance indicator to compare regularity of service between urban bus operators. *Transportation Research Record: Journal of the Transportation Research Board*, 2216(1):33–41, 2011.

B. Tucker. Using wireless location technology to estimate origin destination matrices: A feasibility study. Master's thesis, Purdue University, 2009.

A. Tung, J. Han, L. Lakshmanan, and R. Ng. Constraint-based clustering in large databases. In *Database Theory - ICDT*, pages 405–419. Springer, 2001.

L. Tung, T. Chien, T. Wang, S. Lin, C.and Jeng, and L. Chen. A study of comfort measuring system using taxi trajectories. In *IEEE 17th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 877–882. IEEE, 2011.

S. Turner, W. Eisele, R. Benz, and D. Holdener. Travel time data collection handbook. Technical report, 1998.

M. Turnquist and S. Blume. Evaluating potential effectiveness of headway control strategies for transit systems. *Transportation Research Record*, (746), 1980.

A. Turnquist. Strategies for improving bus transit service reliability. Technical report, Transportation Research Board, 1982.

University of Southampton (UOS). Public transport priority: State of the art review. pricilla project, deliverables 2,, 2002. acessed October 2013.

J. van Lint and N. Van der Zijpp. Improving a travel-time estimation algorithm by using dual loop detectors. *Transportation Research Record: Journal of the Transportation Research Board*, 1855(1):41–48, 2003.

H. van Lint. *Reliable travel time prediction for freeways.* Netherlands TRAIL Research School, 2004.

N. van Oort. *Service reliability and urban public transport design.* Netherlands TRAIL Research School, 2011.

L. Vanajakshi, S. Subramanian, and R. Sivanandan. Travel time prediction under heterogeneous traffic conditions using global positioning system data from buses. *IET intelligent transport systems*, 3(1):1–9, 2009.

J. Vitter. Random sampling with a reservoir. *ACM Trans. Math. Softw.*, 11(1):37–57, March 1985.

N. Vu and A. Khan. Bus running time prediction using a statistical pattern recognition technique. *Transportation Planning and Technology*, 33(7):625–642, 2010.

V. Vuchic. *Urban Transit: Operations, Planning, and Economics.* Wiley, 2005.

Z. Wall. *An algorithm for predicting the arrival time of mass transit vehicles using automatic vehicle location data.* PhD thesis, University of Washington, 1998.

H. Wang, W. Fan, P. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235. ACM, 2003.

F. Wang. Agent-based control for networked traffic management systems. *Intelligent Systems, IEEE*, 20(5):92–96, 2005.

M. Wardman and G. Whelan. Twenty years of rail crowding valuation studies: Evidence and lessons from british experience. *Transport Reviews*, 31:379–398, 2011.

M. Wardman. Public transport values of time. *Transport policy*, 11(4):363–377, 2004.

P. Welding. The instability of a close-interval service. *OR*, pages 133–142, 1957.

B. Williams and L. Hoel. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of Transportation Engineering*, 129(6):664–672, 2003.

I. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2005.

A. Wong. *Travel time prediction model for regional bus transit.* PhD thesis, University of Toronto, 2011.

S. Yan and H. Chen. A scheduling model and a solution algorithm for inter-city bus carriers. *Transportation Research Part A: Policy and Practice*, 36(9):805–825, 2002.

Y. Yan, Q. Meng, S. Wang, and X. Guo. Robust optimization model of schedule design for a fixed bus route. *Transportation Research Part C: Emerging Technologies*, 25:113–121, 2012.

H. Yang and S. Wong. A network model of urban taxi services. *Transportation Research Part B: Methodological*, 32(4):235–246, 1998.

H. Yang, K. Wong, and S. Wong. Modeling urban taxi services in road networks: Progress, problem and prospect. *Journal of Advanced Transportation*, 35(3):237–258, 2001.

B. Yu and Z. Yang. A dynamic holding strategy in public transit systems with real-time information. *Applied Intelligence*, 31(1):69–80, 2009.

B. Yu, Z. Yang, K. Chen, and B. Yu. Hybrid model for prediction of bus arrival times at next station. *Journal of Advanced Transportation*, 44(3):193–204, 2010.

J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang. T-drive: driving directions based on taxi trajectories. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 99–108. ACM, 2010.

J. Yuan, Y. Zheng, X. Xie, and G. Sun. Driving with knowledge from the physical world. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 316–324. ACM, 2011.

J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun. Where to find my next passenger? In *13th ACM International Conference on Ubiquitous Computing (UbiComp)*, 2011.

Y. Yue, Y. Zhuang, Q. Li, and Q. Mao. Mining time-dependent attractive areas and movement patterns from taxi trajectory data. In *17th International Conference on Geoinformatics*, pages 1–6. IEEE, 2009.

M. Zaki, I. Ashour, M. Zorkany, and B. Hesham. Online bus arrival time prediction using hybrid neural network and kalman filter techniques. *International Journal of Modern Engineering Research*, 3(4):2035–2041, 2013.

D. Zhang, N. Li, Z. Zhou, C. Chen, L. Sun, and S. Li. ibat: detecting anomalous taxi trajectories from gps traces. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 99–108. ACM, 2011.

M. Zhang, J. Liu, Y. Liu, Z. Hu, and L. Yi. Recommending pick-up points for taxi-drivers based on spatio-temporal clustering. In *Second International Conference on Cloud and Green Computing (CGC)*, pages 67–72. IEEE, 2012.

G. Zhang. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing*, 50:159–175, 2003.

J. Zhao, M. Dessouky, and S. Bukkapatnam. Distributed holding control of bus transit operations. In *Intelligent Transportation Systems*, pages 976–981, 2001.

J. Zhao, M. Dessouky, and S. Bukkapatnam. Optimal slack time for schedule-based transit operations. *Transportation Science*, 40(4):529–539, 2006.

Y. Zheng, Y. Liu, J. Yuan, and X. Xie. Urban computing with taxicabs. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, pages 89–98. ACM, 2011.

X. Zhou and H. Mahmassani. Dynamic origin-destination demand estimation using automatic vehicle identification data. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):105–114, 2006.

K. Zhu et al. Time-dependent origin-destination estimation: Genetic algorithm-based optimization with updated assignment matrix. *KSCE Journal of Civil Engineering*, 11(4):199–207, 2007.

W. Zhu, K. Boriboonsomsin, and M. Barth. Defining a freeway mobility index for roadway navigation. *Journal of Intelligent Transportation Systems*, 14(1):37–50, 2010.

S. Zolfaghari, N. Azizi, and M. Jaber. A model for holding strategy in public transit systems with real-time information. *International Journal of Transport Management*, 2(2):99–110, 2004.