CHALMERS

moBil Elbilspool

Implementation av driftoptimering genom dynamisk prissättning i en mjukvaruprototyp för friflytande elbilspooler

Kandidatarbete inom Data- och informationsteknik

JESPER LLOYD ROLAND HELLSTRÖM KEYTE
VIKTOR DAHL MARCUS JOHANSSON

Institutionen för Data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2012
Kandidatarbete/rapport nr 2012:69

Abstract

Free-floating carsharing systems allow one-way trips that start and end anywhere within the geographical boundaries of an operating area, most often the central part of a city. As opposed to traditional car rental services, where cars must be returned where they were picked up, vehicles may become poorly distributed. In this project, methods for optimizing operating procedures through dynamic pricing were developed. The goal was to enable a cost effective way of ensuring vehicle availability, and consequently profitability, in free-floating carsharing systems.

Two models for dynamic pricing were developed, and later implemented in a prototype of an IT-solution for free-floating carsharing systems. The basis of the first model was a verified model for cost minimization of station-based carsharing systems with one-way travel. The model is MILP based and is presented in the thesis *Decision support tools for carsharing systems with flexible return time and stations* by Alvina Kek (2006). Difficulties with the integration of dynamic pricing brought about the development of a new model, based on concepts proposed in *Reinventing the Automobile* by William Mitchell et al. (2010).

The results were visualized in an Android application developed for a car-mounted tablet computer. Other software developed include a mobile application for Android aimed for the carsharing system users, as well as essential server-side applications, primarily written in Java.

The experience gained from this project pave the way for full scale implementation of models for dynamic pricing. However, in order to prove an actual cost minimizing effect this type of model may have, both field experiments and behaviour analysis of users are needed. Presented models are sufficiently well-described to allow for future implementation for further analysis and testing.

Sammanfattning

Friflytande bilpooler tillåter envägsresor med start och slut vid vilken parkeringsplats som helst inom ett avgränsat driftområde, vanligen en stadskärna. Till skillnad från traditionella bilpooler, vari bilar alltid ska återlämnas där de hämtats, finns risk att fordon blir olämpligt fördelade. Driftoptimering genom dynamisk prissättning utvecklades i detta arbete med målet att möjliggöra ett kostnadseffektivt sätt att säkerställa tillgänglighet av fordon, och därmed lönsamhet, i friflytande bilpooler.

Två modeller för dynamisk prissättning utvecklades, samt implementerades i en prototyp av en IT-lösning för friflytande elbilspooler. Grunden till den första modellen utgjordes av en verifierad modell för kostnadsminimering av stationsbaserade bilpooler med envägsresor. Modellen är baserad på MILP och hämtades ur Alvina Keks avhandling *Decision support tools for carsharing systems with flexible return time and stations* (2006). Svårigheter i integrationen av dynamisk prissättning föranledde utvecklingen av en ny modell, baserad på koncept från *Reinventing the Automobile* av William Mitchell et al. (2010).

Resultatet visualiserades i en Android-applikation ämnad att köras på en bilmonterad surfplatta. Ytterligare utvecklad mjukvara innefattar en mobilapplikation för Android riktad mot bilpoolens användare, samt essentiella applikationer på serversidan, huvudsakligen skrivna i Java.

Erfarenheterna från detta arbete bereder väg för utvecklingen av fullskaliga implementationer av modeller för dynamisk prissättning. För att kunna påvisa faktisk kostnadsminimerande effekt av dylika modeller krävs dock både fältexperiment och beteendeanalys av användare. Modellerna som utvecklades är tillräckligt välbeskrivna för att efterföljande arbeten utifrån denna rapport ska kunna implementera dem för vidare analys och testning.

Förord

Författarna har under arbetets gång fått ta del av information från ett VINNOVA-finansierat projekt lett av Viktoriainstitutet som berör friflytande elbilspooler. Viktoriainstitutet utför forskning och utveckling inom tillämpad informationsteknik i samarbete med industri, den offentliga sektorn och universitet, med målet att hjälpa svensk bil- och transportindustri driva sin verksamhet för hållbar utveckling och tillväxt (Viktoriainstitutet, 2012).

Resultatet av arbetet, vilket omfattar denna rapport och den implementerade mjukvaruprototypen, ägs och ansvaras för uteslutande av författarna. Medgiven inblick i Viktoriainstitutets projekt har väglett vårt arbete i riktning mot ett praktiskt tillämpbart resultat.

Vi vill därför framföra ett stort tack till de verksamma vid Viktoriainstitutet med vilka vi utbytt idéer, speciellt Henrik Engdahl, Kristoffer Lidström, Johan Wedlin och Mats Williander. Genom detta samarbete har vi delgivits relevanta kunskaper och erfarenheter samt användbar återkoppling på våra resultat och teorier under arbetets gång.

Vi vill även tacka vår handledare Per Zaring som lett oss till många insikter rörande organiserad mjukvaruutveckling samt gjort oss uppmärksamma på vikten av välplanerad kravhantering.

Innehåll

Figurer

Algoritmer

Terminologi och förkortningar

Anställd	En person anställd av bilpoolen i syfte att relokera och underhålla bilpoolens bilar.
Bilflotta	En mängd bilar som en bilpool tillgängliggör för uthyrning.
Bilpool	Ett system för betald användning av bilar under mer flexibla tidsintervall än vanlig biluthyrning. Vanligen hämtas och återlämnas bilarna på speciella stationer.
Bokning	En tidsperiod under vilken en av bilpoolens kunder har exklusiv tillgång till en bil. Under denna period är bilen *bokad*.
CPLEX	En IDE för uppställning och lösning av optimeringsproblem.
CSV	*Comma Separated Values*. Ett enkelt filformat där data lagras i kolumner separerade av kommatecken.
Driftområde	Det område inom vilket bokningar och avlämningar tillåts.
Dynamisk prissättning	Tilldelning av olika priser på samma produkt eller tjänst baserat på en uppsättning faktorer i en prismodell. Används vanligtvis för att maximera vinst.
Friflytande bilpool	En bilpool där bilar kan hämtas och återlämnas på valfri parkeringsplats inom ett driftområde.
GMPL	*GNU Mathematical Programming Language*. Ett specialiserat programspråk till för att beskriva linjära optimeringsproblem.
ITS	*Intelligent transportation system*. Transportsystem som använder informationsteknik för att på olika sätt förbättra drift och tjänster.
OPL	*Optimization Programming Language*. Ett specialiserat programspråk liknande GMPL. OPL används av IBM CPLEX.
Priselasticitet	Mått på hur en förändring av priset på en vara påverkar efterfrågan på produkten. Efterfrågan på produkter med stor priselasticitet sjunker kraftigt som följd av höjda priser.
Relokering	Förflyttning av en bilpools bilar, utförd av personal. Syftet är att förbättra den geografiska fördelningen av bilar med avseende på efterfrågan.
Urban mobilitet	Framkomlighet i en innerstad genom användning av en rad olika tillgängliga färdmedel: Kollektivtrafik, hyrda och privatägda bilar, cyklar etc.

1. Inledning

1.1. Bakgrund

Detta avsnitt fungerar som en introduktion till bilpoolskonceptet i allmänhet och syftar till att redogöra för vad som gör friflytande elbilspooler kommersiellt intressanta. Detta görs genom att först definiera vad en bilpool är, samt vad som åtskiljer friflytande och konventionella bilpooler. Sedan visas hur ITS-lösningar (Intelligent Transport Systems) – speciellt dynamisk prissättning – kan utnyttjas för att öka lönsamheten hos bilpooler, specifikt friflytande sådana.

1.1.1. Personlig mobilitet idag och imorgon

Världens omfattande infrastruktur för framkomst och påfyllning av bränsle erbjuder bilägare en hög grad av rörlighet. Motorfordon för personlig framkomst nyttjas flitigt av människor jorden över och är en förutsättning för mångas levnadssätt. De allra flesta av dessa fordon drivs med fossila bränslen, något som gör tydligt avtryck på miljön (Granovskii et al., 2006).

Förbränningsmotorer har påvisat negativa egenskaper vad gäller avgaser och buller. På senare år har intresset för elbilar därför ökat, då de i regel är både tystare och miljövänligare. Miljömedvetna konsumenter måste dock ha överseende med att dagens elbilar lider av kort räckvidd och lång laddningstid, samtidigt som de är förhållandevis dyra i inköp (Granovskii et al., 2006).

Trots dessa tekniska begränsningar kan elbilen utgöra ett ekonomiskt fördelaktigt alternativ. Konkurrenskraften kommer huvudsakligen från den låga bränslekostnaden, som är mycket lägre än hos bensindrivna bilar (U.S. Department of Energy, 2011). För att vinstmarginalen från det billigare bränslet ska kunna göra en inverkan på den högre inköpskostnaden, måste elbilars nyttjandegrad överstiga den av bilar som drivs på fossila bränslen. Denna marginal skulle kunna ökas med hjälp av så kallade *friflytande bilpooler*[1], varigenom elbilar skulle kunna utnyttjas på ett mer effektivt sätt (Williander, 2011).

[1] Det engelska uttrycket “car pooling” motsvarar svenskans “samåkning”, snarare än “bilpool”. Den korrekta engelska termen för “bilpool” är “carsharing system”.

1.1.2. Vad är en bilpool?

Sveriges första bilpool startades i Lund 1976 och innebar ett sätt för personer som ibland ville hyra bil, men som inte ägde själv, att komma i kontakt med någon som gjorde det. Idag bedrivs bilpooler av både föreningskooperativ och kommersiella aktörer, där pris för medlemskap och uthyrning ofta innefattar centraliserade kostnader för försäkringar och fordonsservice, åligganden som få bilanvändare finner besvärande att fråntas. Upplägget skiljer sig från vanlig biluthyrning på så sätt att bilarna vanligen kan hyras med kort varsel och under relativt korta tidsperioder (Helmersson, 2012).

Konventionella bilpooler kräver i regel att en använd bil lämnas av där den plockades upp, men organisationer som administrerar ett flertal bilpooler kan mycket väl tillåta envägsresor däremellan. I sådana sammanhang kan ingående fordon ses utgöra en och samma bilpool, i vilken en utsedd plats för återlämning benämns en *station*. Likt uthyrningsfirmor som tillåter envägsresor mellan städer, föreläggs det således den administrerande organisationen att säkerställa balans i fördelningen av fordon mellan stationer. Detta är en uppgift som kan vara mycket kostsam för bilpoolen.[2]

1.1.3. Elbilar i friflytande bilpooler

Det som utmärker friflytande bilpooler är avsaknaden av fasta stationer för upphämtning och avlämning. Det kan argumenteras att denna form av bildelning faktiskt ger användarna större frihet än att äga själv, eftersom biltyp kan väljas efter behov vid varje resa. Resonemanget förutsätter dock att lediga bilar till önskad grad finns tillgängliga att möta efterfrågan. Detta ställer krav på både bilflottans storlek samt fördelningen av lediga bilar inom det driftområde vari avlämningar tillåts.

I en friflytande bilpool kan begränsningarna hos dagens elbilar göras avsevärt mindre påtagliga för användarna. Kostnaden för fordonsinköp fördelas på alla användare, och vad gäller räckvidden är den normalt över 16 mil (Granovskii et al., 2006), vilket är tillräckligt för många stadsresor. Om batterinivån ändå når kritiskt låg nivå under körning, är det bara för användaren att leta upp en annan bil i närheten och byta. Det är sedan upp till bilpoolens anställda[3] att se till att bilar som behöver laddas förflyttas till laddplatser. Kostnadseffektiva strategier för att lösa liknande uppgifter är avgörande för systemets lönsamhet (Jensen, 2010).

[2] I fallet med envägsresor mellan städer har konceptet Hertz Freerider utvecklats för att återställa balans i fordonsfördelningen. Biluthyrningsfirman Hertz ger därigenom användare möjlighet att göra vissa resor helt gratis (The Hertz Corporation, 2012).

[3] Notera att "anställd" här används i den specifika mening som anges i rapportens terminologiavsnitt.

1.1.4. ITS, driftoptimering och dynamisk prissättning

I det fåtal storskaliga friflytande elbilspooler i bruk idag utnyttjas ITS-lösningar för att åstadkomma effektiv och säker förvaltning. Åtkomst till en bil kan smidigt begränsas till användare med giltig bokning, samtidigt som övervakning av fordonsflottan, hantering av kundregister och bokningar samt debitering kan förenklas avsevärt. Sådana system kan utökas att omfatta kostnadsminimerande algoritmer[4] samt beslutsunderlag för bilpoolsanställda, men i vilken utsträckning det görs idag är inte allmänt känt.

Metoder för driftoptimering av friflytande elbilspooler som baseras på principen om *dynamisk prissättning* har föreslagits av forskare vid Massachusetts Institute of Technology (MIT)[5] (MIT Smart Cities Group, 2010). Många olika former av dynamisk prissättning kan observeras idag, exempelvis tågbiljetter som blir dyrare när avgången närmar sig, eller elpriser som justeras för att jämna ut svängningar i efterfrågan. Dylika flexibla prismodeller är i allmänhet användbara när variationer förväntas i balansen mellan tillgång och efterfrågan av en vara. I friflytande bilpooler kan denna mekanism utnyttjas för att ge användarna incitament att agera efter vad som är gynnsamt för systemet. Exempelvis skulle dynamisk prissättning kunna innebära ett kostnadseffektivt sätt att upprätthålla balans i fordonsfördelningen (Mitchell et al., 2010).

1.2. Problemformulering

Direkt tillämpbara modeller för driftoptimering i friflytande elbilspooler finns, i den mån de utvecklats vid tiden för detta arbete, dessvärre inte fritt tillgängliga. Samtidigt kan de vara nyckeln till kommersiellt gångbara friflytande elbilspooler.

1.3. Syfte

Denna rapport syftar till att redogöra för utvecklingen av en prototyp av en IT-lösning för friflytande elbilspooler, med fokus på driftoptimering genom dynamisk prissättning. Vidare syftar rapporten till att redogöra för hur ett par modeller för dynamisk prissättning utformades och implementerades i denna mjukvaruprototyp.

1.4. Avgränsningar

I det här avsnittet redogörs för vilka närliggande områden, relaterade till modeller för dynamisk prissättning, som legat utom ramarna för detta arbete. Vidare redogörs för vilka delar av mjukvaruutvecklingen som bortsågs från eller nedprioriterades.

[4] Ett exempel är Drop Car från Good Travel Software.

[5] MIT-forskarna har en vision om att elbilar ska utgöra en integrerad del av framtidens smarta elnät, exempelvis genom att erbjuda tillfällig lagring av överskottselektricitet (Mitchell et al., 2010).

1.4.1. Utveckling av modeller för dynamisk prissättning

Under utvecklingen av modeller för dynamisk prissättning togs enbart hänsyn till de aspekter av bilpoolsdrift som hade en direkt inverkan på utformningen av mjukvaruprototypen. Inom projektet gjordes därför exempelvis inget försök att operationalisera kunders reaktioner på dynamiska priser.

1.4.2. Implementation av mjukvaruprototypen

Implementationen av mjukvaruprototypen utfördes främst i demonstrativt syfte och därför prioriterades den funktionalitet som ansågs särskilt relevant för det undersökta området.

Säkerhet och testning

Prototyputvecklingen har fokuserat på funktionalitet snarare än säkerhet. Ett heltäckande driftsystem för friflytande elbilspooler skulle omfatta bland annat funktionalitet för debitering och fakturering, vilket skulle ställa höga krav på säkerhet och robusthet. Då säkerhetsaspekten skulle påverkat grundläggande designval och begränsat friheten av det evolutionära tankesättet ansågs det inte berättigat att ägna tid åt det.

Då mjukvaran främst utvecklades i demonstrativt och undersökande syfte genomfördes inga användartester. Utöver det ledde frånvaron av både en fast kravbild och kännedom om förväntat beteende till att strukturerad enhetstestning bedömdes ovidkommande.

Användargränssnitt

Av de användargränssnitt som skulle kunna utvecklas mot servern har projektet begränsats till implementation av Android-applikationer för mobiltelefoner och surfplattor avsedda att användas av slutanvändaren.

Då tanken med en friflytande bilpool är att en kund ska kunna hitta och boka bilar oavsett tid och plats, ansågs det att denna funktionalitet borde införas i smarta mobiltelefoner, med anledning av möjligheten att utnyttja inbyggda GPS-mottagare. På grund av svårigheten att få åtkomst till sådan inbyggd funktionalitet genom rena webbapplikationer valdes detta alternativ bort.

Vad gäller grafiska operatörsgränssnitt så ansågs det inte vara värt att utarbeta dylika, då serverfunktioner antogs komma att förändras frekvent under utvecklingsfasen. All interaktion på operatörs- och administratörsnivå utfördes således genom text-baserade gränssnitt.

1.5. Rapportens upplägg

I denna rapport separeras metodbeskrivningen från presentationen av genomförandet på så sätt att redogörelser för metodval huvudsakligen sker i kapitel 2, *Metod*. Där diskuteras svårigheter med att finna beprövade metoder för utveckling av modeller för dynamisk prissättning i friflytande bilpooler. Där redogörs även för vissa av de riktlinjer som tillämpats vid mjukvaruutvecklingen.

Avsnitt 3.2, *Driftoptimering i friflytande elbilspooler: En introduktion*, och 3.3, *Driftoptimering för stationsbaserade bilpooler: Kek-modellen*, redogör för den teoretiska grund på vilken arbetet med utveckling av modeller för dynamisk prissättning baserades.

Eftersom projektet består av både ett omfattande teoretiskt arbete, samt praktiskt implementation av mjukvara, har beskrivningarna av dessa delars respektive genomförande delats i två kapitel. Av samma anledning redogörs resultaten för i två olika kapitel.

Kapitel 4, *Genomförande: Utveckling av modeller för dynamisk prissättning*, behandlar det teoretiska arbete som projektet bestod av, medan kapitel 5, *Genomförande: Utveckling av mjukvaruprototyp*, behandlar den praktiska implementationen som grundar sig i det teoretiska arbetet.

I kapitel 6, *Resultat: Utvecklade modeller för dynamisk prissättning*, presenteras de två modeller för dynamisk prissättning som utvecklades: Modell 1 (Kek-baserad) och modell 2 (fordonsbalans-baserad). Där återfinns detaljerade beskrivningar av de modeller och algoritmer som legat till grund för den dynamiska prissättningen. Resultatet av mjukvaruutvecklingen presenteras i kapitel 7, *Resultat: Utvecklad mjukvaruprototyp*, där den färdigställda mjukvarans funktionalitet beskrivs översiktligt. Diskuterande resonemang kring dessa resultat presenteras sedan i kapitel 8, *Diskussion*.

Avslutningsvis redovisas, i kapitel 9, *Slutsatser*, hur väl resultatet kan anses bemöta de problemställningar som föranledde detta arbete. På detta följer riktlinjer för framtida arbeten.

2. Metod

Inledningsvis genomfördes en förstudie där rapporter och avhandlingar på området driftoptimering i bildelningssystem studerades. Målet var att utforska vilka metoder som fanns tillgängliga för att effektivisera driften av bilpooler, och vilken roll dessa metoder skulle kunna ha vid utveckling av dynamisk prissättning. Därefter utfördes en översiktligt undersökning av användningsvillkor och prismodeller för kommersiella bildelningssystem, med varierande omfång och bilantal. Syftet var att ge en ungefärlig bild av vilka prismodeller som kunde ligga till grund för, eller jämföras med, dynamisk prissättning.

Då förstudien fastslog att direkt applicerbara modeller för driftoptimering i det specifika fallet friflytande bilpooler inte stod att finna, beslutades det att sådana modeller skulle utarbetas för att implementationen skulle vara meningsfull. Arbetet har därför haft två inriktningar, en bestående av utveckling av teoretiskt underlag för driftoptimering, och den andra implementation av mjukvaruprototypen baserat på det teoretiska underlaget.

2.1. Teoriutveckling

Då tidigare forskning inom driftoptimering i friflytande bilpooler saknas, saknas också etablerade metoder och kriterier för utveckling av modeller för detta. Därmed antog det teoretiska arbetet en utforskande karaktär.

Först utarbetades prismodeller lämpade för dynamisk prissättning i det friflytande elbilspoolskonceptet. Därefter undersöktes möjligheten att integrera dynamisk prissättning i en känd modell för optimering av stationsbaserade bilpooler, i denna rapport benämnd *Kek-modellen*. En undersökning gjordes sedan kring möjligheten att 1) överföra denna modell till det friflytande konceptet och 2) modifiera modellen genom att införa dynamisk prissättning. Det visade sig dock att dessa modifikationer begränsade modellens praktiska tillämpbarhet, vilket föranledde utvecklingen av en enklare modell för dynamisk prissättning: den *fordonsbalans-baserade*. Denna enklare modell användes sedan i den slutgiltiga implementationen.

2.1.1. Kriterier och utgångspunkter för modeller

Utvärdering av hur väl olika metoder för driftoptimering faktiskt fungerar skulle kräva antingen testning i verkliga system, eller utvärdering genom simulering. Då det inte var

rimligt att utföra testning inom tidsramen för projektet och simuleringsverktyg anpassade för uppgiften inte fanns att tillgå, sattes inga konkreta mätbara kriterier för effektivitet. Likväl fanns tydliga målsättningar gällande vilka olika aspekter av drifthantering som de två modeller som utvecklades ämnade effektivisera.

Den första modellen utgick från den modell av stationsbaserade bilpooler som presenteras i Kek (2006). Keks modell utgick från att definiera ett antal kostnader relaterade till förflyttning och underhåll av bilar, samt konsekvenser av tomma eller överfulla stationer. I det här arbetet modifierades Keks modell genom införsel av variabler och beroenden för att inkludera dynamisk prissättning direkt i modellen.

Den andra modellen, den fordonsbalans-baserade, byggde till stor del på egna idéer, men utgick till viss del från rörande visualisering av skillnader i tillgång och efterfrågan i form av värmekartor som framförs i Mitchell et al. (2010). Denna utgångspunkt nödvändiggjorde utvecklandet av modeller för tillgång och uppskattad efterfrågan. Tillgångsmodellen utvecklades baserat på egna antaganden, emedan modellen för uppskattad efterfrågan delvis utgår från principerna för cykliska användningsmönster som används i Kek (2006, s. 25). Snarare än att direkt minimera kostnader syftar denna modell endast till att uppnå hög tillgänglighet av lediga bilar. Att försöka uppnå hög tillgänglighet görs dock med avsikten att det i förlängningen ska leda till lägre driftkostnader och högre inkomster för bilpoolen.

2.2. Mjukvaruutveckling

Mjukvaruutvecklingen har varit underordnad det teoretiska arbetet på så sätt att mjukvaruprototypen främst utgjort en demonstrationsplattform för idéutveckling. Arbetet följde ett urval av agila metoder hämtade från RUP (Rational Unified Process) (Sommerville, 2010, s. 50), och använde en evolutionär struktur för att lättare kunna hantera en föränderlig kravbild (Sommerville, 2010, ss. 235–239).

Utvecklingen skedde i tre iterationer och till stor del parallellt med teoriutvecklingen. I den första iterationen implementerades driftoptimering i enlighet med den ursprungliga Kek-modellen, och den modifierade modell som tog hänsyn till dynamisk prissättning. I den andra iterationen utvidgades mjukvaruprototypen för att tillhandahålla grundläggande funktionalitet för bilpooler, inräknat ett bokningssystem, och en kundinriktad mobilapplikation. I den tredje och sista iterationen implementerades en enklare modell för dynamisk prissättning, samt en Android-applikation för en bilmonterad surfplatta.

Den färdiga mjukvaruprototypen presenterades slutligen vid ett anförande på Viktoriainstitutet, som sedan använde prototypen i en demonstrationsfilm för det friflytande elbilspoolskoncept de utvecklat.[1]

[1] Viktoriainstitutets koncept är känt som *BilDelaren*. Demonstrationsfilmen finns att tillgå på `http://youtu.be/3Jv92i2erIY`.

3. Teoretisk bakgrund

Detta kapitel redogör för utgångsläget för det teoretiska arbete som presenteras i denna rapport. Den information som presenteras i kapitlet är i huvudsak resultat av förstudien och utgör således i regel sammanfattande beskrivningar av redan utvecklade koncept. Koncept och begrepp rörande bilpooler och driftoptimering ges här ett sammanhang som underlättar förståelsen för senare resonemang, och som tydliggör avgränsningen mellan redan känd information och nyvunnen sådan.

3.1. Relaterade projekt

Inga tidigare publicerade arbeten med syfte som sammanfaller med eller omfattar detta arbetes syfte har påträffats av författarna. Ej heller har författarna påträffat öppet publicerade studier som behandlar området dynamisk prissättning för friflytande bilpooler. Projekt med liknande principer, och som ansetts nära relaterade till området, har dock påfunnits. Nedan sammanfattas några av dessa relaterade arbeten. Urvalet består av projekt som endera behandlar ämnen mycket nära knutet till detta projekts syfte, eller som har bidragit med insikter kring driftoptimering av friflytande bilpooler.

Smart Cities Research Group är en forskningsgrupp vid MIT Media Lab som bland annat bedriver forskning inom ett område de benämner *smart fleet management solutions*. SCRG driver projektet Mobility-on-Demand som är en lösning för urban mobilitet som innefattar både dynamisk prissättning och utveckling av koncept för friflytande elbilspooler. Projektet drar paralleller till det principiellt närliggande forskningsområdet om lastutjämning i elnät för att resonera kring dynamisk prissättning (MIT Smart Cities Group, 2010).

Drop Car är ett projekt som drivs av en forskningsgrupp på Trinity College Dublin i samarbete med företaget Good Travel software (Distributed Systems Group, 2012). Projektet ämnar utveckla mjukvara som tillgängliggör användningen av dynamisk prissättning för existerande bilpooler, med stöd för både envägsresor och friflytande system. Det faktum att projektet är kommersiellt tyder på att det finns intresse för den typ av lösningar som presenteras i denna rapport. På grund av dess kommersiella natur finns i skrivande stund inte någon utbredd information om hur den dynamiska prissättningen implementeras. Drop Car stöds och finansieras i viss mån av Enterprise Ireland. Mjukvaran beräknas släppas på marknaden under 2012.

ICVS (Intelligent Community Vehicle System), även känt under namnet Honda Diracc, var ett projekt som använde optimeringsalgoritmer för drifthantering av en bilpool med flexibla återlämningstider och återlämningsplatser. Projektet drevs av Honda Motor Company, men lades ned 2008 på grund av svårigheter med att upprätthålla en hög kvalitet på tjänsten när antalet medlemmar ökade (Tan, 2008).

Vélib' är en tjänst för cykeldelning som drivs i Paris. Användare av tjänsten kan hämta och lämna cyklar på ett stort antal stationer inom staden. Vélib' kan klassificeras som en stationsbaserad cykelpool med envägsresor, och delar därmed de problem med fördelning av fordon som återfinns i bilpooler.

car2go är en organisation som driver friflytande bilpooler på ett antal ställen i världen, däribland Ulm, Austin och Vancouver. car2go stöds av den tyska biltillverkaren Daimler (Garthwaite, 2011). De resonemang om dynamisk prissättning som legat till grund för denna rapport har, för att vara givande, tvunget knutit an till praktiska omständigheter. Även om car2Go inte använder dynamisk prissättning har systemet, som enda betydande kommersiella friflytande bilpool, varit användbart som utgångspunkt för dylika resonemang. Vidare var car2go en av de huvudsakliga inspirationskällorna som ledde till idén bakom detta projekt.

3.2. Driftoptimering i friflytande elbilspooler: En introduktion

Detta avsnitt syftar till att utförligt beskriva vissa egenskaper hos friflytande elbilspooler som riskerar medföra betydande driftrelaterade kostnader. Härigenom ges den teoretiska grund för de ansatser till driftoptimering som gjorts i detta arbete. Begrepp såsom balanserad fordonsfördelning, relokering av fordon samt dynamisk prissättning presenteras tillsammans med begreppens konkreta betydelser i denna rapport.

3.2.1. Driftrelaterade processer med utrymme för optimering

Det finns ett antal olika aspekter av friflytande elbilspooler som skulle kunna effektiviseras genom någon form av processoptimering. Exempel på detta är inköps- och avsäljningsstrategi för bilflottan, hur och när fordon underhålls samt utbyggnad av infrastrukturen för laddning[1].

I detta arbete fokuseras på optimeringen av den särskilda process som besvarar utmaningen att garantera tillgänglighet av fordon i ett system där det är omöjligt att exakt avgöra hur bilar kommer att fördela sig. Denna sorts optimering är högst relevant, då tillgängligheten är något som är direkt avgörande för systemets lönsamhet (Jensen, 2010).

[1] Dagens laddstolpar kan i framtiden komma att kompletteras av markbundna plattor för induktionsladdning (Karlberg, 2011).

3.2.2. Obalanserad fordonsfördelning som konsekvens av envägsresor

Bilpooler som tillåter envägsresor öppnar för situationer där bilflottan kan bli ojämnt fördelad med avseende på efterfrågan (Nair, 2010, s. 10), så till den grad att användares behov inte tillfredsställs i önskad utsträckning. I denna rapport benämns en sådan fordonsfördelning som *obalanserad*.

Tillfällig obalans kan vara acceptabelt då systemet potentiellt kan återgå till ett balanserat tillstånd inom rimlig tid. Föreligger skäl att anta att så inte kommer ske, kan det mest lönsamma alternativet vara att manuellt flytta bilar för att uppnå en mer balanserad fordonsfördelning. I denna rapport används termen *relokering* för att beskriva en sådan balanserande förflyttning av fordon.

3.2.3. Metoder för balansering av fordonsfördelning

System för delning av cyklar, exempelvis Vélib', ställs inför liknande utmaningar som friflytande bilpooler gällande upprätthållande av balans i fordonsfördelning. Det finns dock effektiva metoder för att omfördela cyklar: Olämpligt placerade cyklar samlas upp på lastfordon för att sedan omfördelas över staden. Detta är en kostnadseffektiv lösning på problemet då flera cyklar kan fraktas samtidigt. Exempelvis använder Vélib' specialiserade släpvagnar för omfördelning av cyklar (Transport Canada, 2012). Bilar är dessvärre opraktiska att samfrakta på liknande vis och måste därför i praktiken framföras för egen maskin.

Eftersom manuell relokering av elbilar är en relativt kostsam process, där en anställd endast kan flytta ett fordon åt gången,[2] finns skäl att minimera den tid som ägnas åt det. Då elbilar förr eller senare behöver laddas, tvättas eller repareras kan relokeringar utföras i samband med detta. Det är dock inte säkert att relokeringar enbart i samband med underhåll är tillräckligt för att upprätthålla balans i alla friflytande bilpooler.

För att maximalt utnyttja en given fordonsflotta kan det vara fördelaktigt att ta hjälp av användarna av bilpoolen vid upprätthållandet av balans. Användare har i allmänhet ett intresse av att balans upprätthålls, då det leder till bättre tillgänglighet på bilar. Användare lär därmed kunna motiveras att agera så som gynnar systemet, så länge det inte skiljer sig avsevärt från användarens önskade beteende. Ytterligare incitament skulle kunna införas för att få användare att justera sitt beteende. Rätt utformade incitament skulle kunna utgöra grunden för en kostnadseffektiv balanseringsstrategi (Mitchell et al., 2010).

[2] *Virtual towing* är en intressant framtida lösning som ger en anställd möjligheten att flytta flera bilar samtidigt (Mitchell et al., 2010).

3.2.4. Självbalansering genom dynamisk prissättning

I detta arbete används *dynamisk prissättning* som benämning på ständigt uppdaterade prisincitament, utformade för att maximera den balanserande effekten av användares resor. Konceptet bygger på antagandet att en användare söker minimera kostnaden för sin resa (Mitchell et al., 2010). Den grundläggande utmaningen med att utforma en dynamisk prissättningsmodell för friflytande elbilspooler innebär således att besvara två frågor: Hur definieras måttet på balans i en fordonsfördelning på ett användbart sätt, samt hur bör incitament utformas så att användares prismedvetenhet leder dem att vilja maximera detta balansmått?

En enkel ansats till ett användbart balansmått skulle, för enkelhetens skull, kunna ta hänsyn enbart till tillgänglighet utan att direkt blanda in lönsamhet[3]. En fördelning skulle då till exempel kunna betraktas balanserad om den erbjuder ett fordon inom 500 m från varje användare som är i behov av bil. En sådan definition väcker ytterligare en intressant frågeställning, nämligen hur man avgör var det är sannolikt att behov uppstår.

3.2.5. Att förutspå efterfrågan

För att kunna avgöra var fordon bör vara placerade för att tillfredsställa efterfrågan krävs en metod för att avgöra var användare befinner sig då de behöver bilar. Det är rimligt att anta att denna efterfrågan inte är konstant, utan varierar med tiden. *Demand prediction*[4] är det begrepp som i sammanhanget sammanfattar processen att förutse efterfrågan. Resultatet av denna process benämns *förutspådd efterfrågan* i denna rapport.

Att förutspå efterfrågan kan vara så enkelt som att göra kvalificerade gissningar utifrån någon heuristik. Exempelvis kan det antas att användare av friflytande elbilspooler utgör ett representativt tvärsnitt av befolkningen. Man skulle då kunna uppskatta var användare befinner sig genom att studera var människor i allmänhet uppehåller sig.[5] Hur stort behovet är hos en genomsnittlig användare skulle därefter exempelvis kunna uppskattas baserat på tiden på dygnet.

Man skulle också kunna tänka sig mer avancerade metoder för att förutspå efterfrågan som sannolikt ger en ännu bättre uppskattning av framtida efterfrågan. Givet att efterfrågan på bilar uppvisar periodiskt beteende kan man använda sig av historisk data från den faktiska bilpoolen för att förutspå hur efterfrågan i framtiden kommer arta sig. Det gäller då att först modellera precis hur historien kan tänkas upprepa sig, det vill säga hitta en regel som sållar ut den historiska data som är relevant, för att sedan analysera denna med hjälp av statistiska metoder[6]. Det är denna metod för demand prediction som främst används i detta arbete.

[3] I någon mening är tillgänglighet en förutsättning för lönsamhet. Distinktionen görs för att ge ett enkelt exempel utan att behöva elaborera ytterligare i detta samspel.

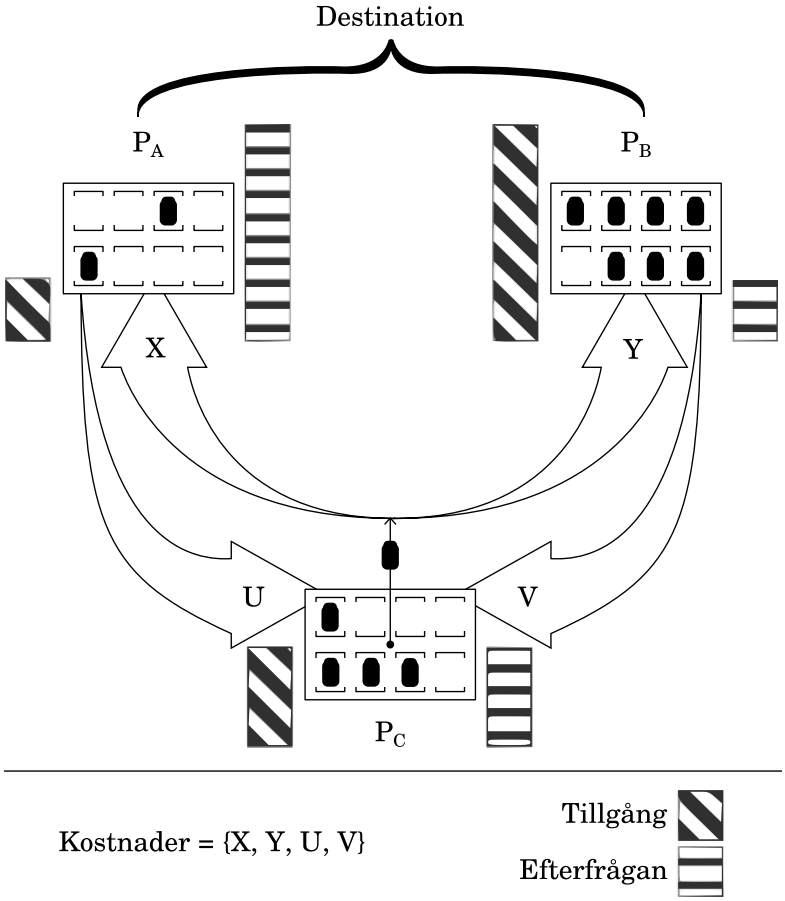
[4] En alternativ benämning är *demand forecasting*.

[5] Tjänsten SpotRank erbjuder information i realtid om geografiska positioner hos ett stort antal smartphone-användare (Skyhook, 2012).

[6] Att hitta mönster i stora mängder data brukar benämnas *data mining*, eller *informationsutvinning*.

3.2.6. Grundläggande princip bakom dynamisk prissättning

Den allmänna modell av dynamisk prissättning som utforskats i detta arbete tar endast hänsyn till var bilar plockas upp och var de lämnas av. Hur de används däremellan är mindre intressant.[7] Prisdifferentiering appliceras därmed endast på just upphämtning och avlämning av bilar beroende på position. Detta görs så att användare uppmuntras att ta bilar från områden med förhållandevis låg efterfrågan i jämförelse med antal tillgängliga bilar, samt lämna bilar där efterfrågan är relativt hög i förhållande till antal tillgängliga bilar (Mitchell et al., 2010). Principen illustreras i figur 3.1.



Figur 3.1. Principen bakom dynamisk prissättning. En resa från P_c till *Destination* kan avslutas vid antingen P_a eller P_b. Brist på bilar vid P_a, medför att resan X görs billigare än Y för att uppmuntra kunden att balansera systemet.

[7] I Mitchell et al. (2010) presenteras konceptet om dynamisk prissättning av olika rutter. Detta kan användas för att jämna ut trafikflödet och därmed minska risken för trafikstockningar samt minska transporttider i allmänhet, men är inte direkt kopplat till bilpoolens lönsamhet.

3.3. Driftoptimering för stationsbaserade bilpooler: Kek-modellen

Detta avsnitt sammanfattar den linjärprogrammeringsmodell som använts som grund vid utveckling av den modell för dynamisk prissättning som behandlas i avsnitt 6.1. Modellen är hämtad från en avhandling av Alvina Kek (2006), och valdes på grund av att den applicerats, med gott resultat, på en riktig bilpool i Singapore (Kek et al., 2009, s. 128).

3.3.1. Modellens syfte och användning

Keks modell utvecklades för att möjliggöra driftoptimering av stationsbaserade bilpooler med envägsresor. Modellen är ett resultat av en av de första vetenskapliga undersökningarna som utförts för att ta fram riktlinjer för drift av sådana bilpooler (Kek et al., 2009). Optimeringen syftar till att minimera bilpoolens driftkostnad och tar hänsyn till kostnaderna för relokering av bilar, förflyttning av personal, underhållsarbeten, löner samt uteblivna inkomster till följd av avvisade kunder. En fullföljd optimering tilldelar värden till en uppsättning beslutsvariabler, vilka sedan kan användas till att ta fram lämpliga riktlinjer för driften av bilspoolen.

Keks modell har tillämpats på en bilpool i Singapore. Det visade sig att de framtagna riktlinjerna – arbetsscheman för relokeringar, underhållsarbete och förflyttningar mellan stationer – minskade bilpoolens driftkostnader och förbättrade bilarnas allmänna tillgänglighet (Kek et al., 2009, s. 158).

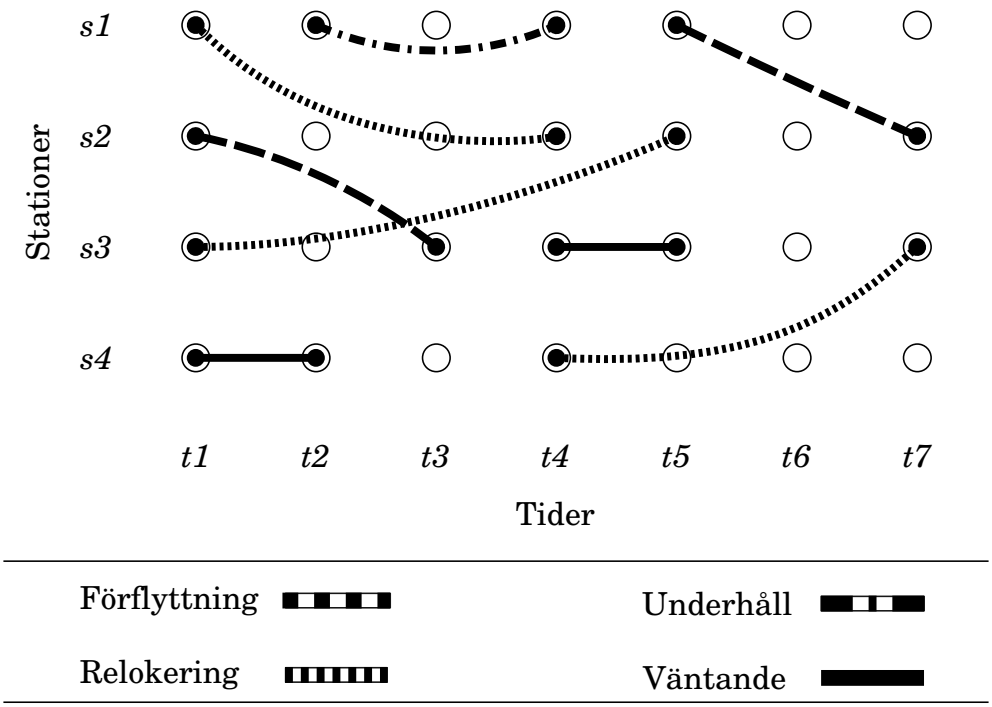
Modellen är formulerad som ett MILP-problem (Mixed Integer Linear Programming), och kan överföras till ett matematiskt programmeringsspråk (såsom redovisas i avsnitt 6.1).

Bilpoolens drift optimeras genom att optimala värden på driftparametrar hittas enligt följande metod: Först definieras ett utgångsläge med ett visst antal bilar, en fordonsfördelning och en utplacering av anställda. Utifrån en fördefinierad tidsperiod, indelad i ett antal tidssteg, provar optimeringsalgoritmen sedan olika kombinationer av handlingar för de anställda i varje tidssteg. Modellen optimeras upprepade gånger för ett stegvis minskat antal anställda, i syfte att finna den optimala arbetsstyrkan. Hela denna procedur görs med hänsyn till den totala kostnaden för att driva bilpoolen, där den mest kostnadseffektiva kombinationen av beslutsvariabler söks. Denna kombination används sedan till att ta fram riktlinjer för driften av bilpoolen.

Som en konsekvens av att optimeringen sker över en fördefinierad tidsperiod begränsas resultatet till indikationer om hur bilpoolen bör drivas under just den valda perioden. Metoden för demand prediction i denna modell utgår därför ifrån att pålitliga värden för bilbehov finns att tillgå, för varje station och tidssteg. Exempelvis skulle ett medelvärde för efterfrågan på en viss veckodag kunna beräknas utifrån historisk användningsdata, för att användas som värde för framtida efterfrågan. Riktlinjer skulle sedan erhållas för hur man skulle driva bilpoolen på denna veckodag.

3.3.2. Matematisk formulering

I modellen sker de anställdas aktiviteter (relokering, förflyttning, eller underhåll) mellan noder i ett nätverk av tid-stationspar, som illustreras i figur 3.2. Denna mängd av aktiviteter benämns i detta avsnitt som A_n, där n betecknar den aktivitet som utförs. $n = 1$ betecknar väntande vid en station, $n = 2$ betecknar en underhållsoperation, $n = 3$ betecknar en förflyttning (utan att relokera), och $n = 4$ betecknar en relokering. De binära beslutsvariabler som avgör om en sådan aktivitet äger rum eller ej, är definierade över mängden av bågar V mellan dessa noder.



Figur 3.2. Aktiviteter representerade som ett tid-stationsnätverk, där varje båge representerar en aktivitet och varje nod representerar en viss station vid en viss tidpunkt. Bågen $(s_4, t_4) \rightarrow (s_3, t_7)$ representerar alltså en anställd som relokerar en bil från station s_4 till s_3 under loppet av tre tidssteg.

Följande beslutsvariabler används i modellen:

t_m	Uppskattad tid för underhåll av en bil
t_{ij}	Uppskattad tid det tar att en köra bil från station i till j
x^k	Binär variabel som för varje anställd k säger om denne jobbar överhuvudtaget
$y^k_{i_t i_{t+1}}$	Binär variabel som säger om anställd k väntar på station i från tidssteg t till $t+1$
$z^k_{i_t i_{t+t_m}}$	Binär variabel som säger om anställd k utför underhåll på en bil på station i från tidssteg t till $t+t_m$
$u^k_{i_t i_{t+t_m}}$	Binär variabel som säger om anställd k förflyttar sig (utan att relokera) från station i till j, mellan tidsstegen t och $t+t_{ij}$
$v^k_{i_t j_{t+t_{ij}}}$	Binär variabel som säger om anställd k relokerar en bil från station i till j, mellan tidsstegen t och $t+t_{ij}$
$d^r_{i_t}$	Heltalsvariabel som säger hur många kunder som nekats att hämta bilar från station i mellan tidpunkterna $t-1$ och t
$s^r_{i_t}$	Samma som d_r fast för avlämning istället för upphämtning
r_{i_t}	Beroende variabel som representerar antalet tillgängliga bilar vid station i vid tidssteg t
$\bar{r}_{i_t}$	Beroende variabel som representerar antalet otillgängliga (på grund av underhåll) bilar vid station i vid tidssteg t.

Utöver beslutsvariablerna består modellen av ett antal inlästa konstanter som utgör uppskattningar av exempelvis operatörskostnader och kostnader att avvisa kunder. Dessa är:

c_{ij}	Kostnad för en anställd att köra en bil från station i till j
c_x	Kostnad för att använda en anställd under optimeringsintervallet
c_d	Kostnaden att neka en kund att hämta en bil
c_s	Kostnaden att neka en kund att avlämna en bil
r_{i_0}	Antal tillgängliga bilar vid station i vid tiden $t=0$
$\bar{r}_{i_0}$	Antal otillgängliga bilar (bilar som är bortplockade för underhållsarbeten) vid station i vid tiden $t=0$
d_{i_t}	Förutspådd efterfrågan för upphämtning vid station i från tidssteg $t-1$ till t
s_{i_t}	Förutspådd efterfrågan för avlämning vid station i från tidssteg $t-1$ till t
m_{i_t}	Förutspått antal bilar i behov av underhåll återlämnade till station i från tidssteg $t-1$ till t
p_i	Antal parkeringsplatser vid station i

I målfunktionen (3.1) summeras alla kostnader som är associerade med avvisning av kunder, samt kostnader för anställda och deras arbetsuppgifter.

$$\begin{aligned} & c_{ij}\left(\sum_{(i_t,j_{t+t_{ij}})\in A_3}\sum_{k\in K} u^k_{i_t j_{t+t_{ij}}} + \sum_{(i_t,j_{t+t_{ij}})\in A_4}\sum_{k\in K} v^k_{i_t j_{t+t_{ij}}}\right) + \\ & + c_x\sum_{k\in L} x^k + c_d\sum_{i_t\in V} d^r_{i_t} + c_s\sum_{i_t\in V} s^r_{i_t} \end{aligned} \tag{3.1}$$

För målfunktionen finns ett antal bivillkor som begränsar lösningarna till att följa ett antal logiska regler. Dessa bivillkor beskrivs av ekvationerna (3.2)–(3.17).

Bivillkor (3.2) ser till att $x^k = 1$ om anställd k är aktiv under det skift som optimeras för. Villkoret ser även till att varje anställd inte kan påbörja fler än en handling vid tidssteg $t = 1$. (3.3) uttrycker att varje anställd, vid ett tidssteg, påbörjar lika många handlingar som denne avslutade vid föregående tidssteg. Bivillkor (3.4) och (3.5) uppdaterar variablerna r_{i_t} och $\bar{r}_{i_t}$ vid varje tidssteg beroende på antalet bilar som avlämnas, upphämtas, och tas in/ut för underhåll. (3.6) säger att antalet bilar på en station inte får överskrida stationens kapacitet. (3.7) säger att man inte kan avvisa fler kunder än antalet som vill ha bilar vid en given tidpunkt. (3.8) till (3.12) säger att variablerna, som uttrycker huruvida en anställd utför en viss handling eller ej, ska vara binära. (3.13) till (3.17) säger att beslutsvariablerna som relaterar till efterfrågan och bilantal inte får vara negativa.

$$\sum_{i\in N} y^k_{i_1 i_2} + \sum_{i\in N} z^k_{i_1 i_{1+t_m}} + \sum_{\substack{i,j\in N\\ i\neq j}} u^k_{i_1 j_{1+t_{ij}}} + \sum_{\substack{i,j\in N\\ i\neq j}} v^k_{i_1 j_{1+t_{ij}}} = x^k \quad \forall k \in L \tag{3.2}$$

$$\begin{aligned} & y^k_{i_{t-1} i_t} + z^k_{\substack{i_{t-t_m} i_t\\ (i_{t-t_m},i_t)\in A_2}} + \sum_{(j_{t-t_{ji}},i_t)\in A_3} u^k_{j_{t-t_{ji}} i_t} + \sum_{(j_{t-t_{ji}},i_t)\in A_4} v^k_{j_{t-t_{ji}} i_t} - y^k_{i_t i_{t+1}} - \\ & - z^k_{i_t i_{t+t_m}} - \sum_{(i_t,j_{t+t_{ij}})\in A_3} u^k_{i_t j_{t+t_{ij}}} - \sum_{(i_t,j_{t+t_{ij}})\in A_4} v^k_{i_t j_{t+t_{ij}}} = 0 \quad \forall i_t \in V, k \in L, t \neq 1 \end{aligned} \tag{3.3}$$

$$\begin{aligned} & r_{i_t} = r_{i_{t-1}} + \sum_{(j_{t-t_{ji}},i_t)\in A_4}\sum_{k\in L} v^k_{j_{t-t_{ji}} i_t} - \sum_{(i_t,j_{t+t_{ji}})\in A_4}\sum_{k\in L} v^k_{i_t j_{t+t_{ji}}} + \\ & + \sum_{\substack{(i_{t-t_m},i_t)\in A_2\\ k\in L}} z^k_{i_{t-t_m} i_t} + (s_{i_t} - s^r_{i_t}) - (d_{i_t} + vd_{i_t} - d^r_{i_t}) - m_{i_t} \quad \forall i_t \in V \end{aligned} \tag{3.4}$$

$$\bar{r}_{i_t} = \bar{r}_{i_{t-1}} - \sum_{k\in L} z^k_{i_t i_{t+t_m}} + m_{i_t} \quad \forall i_t \in V \tag{3.5}$$

$$r_{i_t} + \bar{r}_{i_t} \leq p_i \quad \forall i_t \in V \tag{3.6}$$

$$d^r_{i_t} \leq d_{i_t} + vd_{i_t} \quad \forall i_t \in V \tag{3.7}$$

$$s^r_{i_t} \leq s_{i_t} \quad \forall i_t \in V \tag{3.8}$$

$$x^k \in 0, 1 \quad \forall k \in L \tag{3.9}$$

$$y^k_{i_t i_{t+1}} \in 0, 1 \quad \forall (i_t, i_{t+1}) \in A_1, k \in L \tag{3.10}$$

$$z^k_{i_t i_{t+m}} \in 0, 1 \quad \forall (i_t, i_{t+t_m}) \in A_2, k \in L \tag{3.11}$$

$$u^k_{i_t j_{t+t_{ij}}} \in 0, 1 \quad \forall (i_t, j_{t+t_{ij}}) \in A_3, k \in L \tag{3.12}$$

$$u^k_{i_t j_{t+t_{ij}}} \in 0, 1 \quad \forall (i_t, j_{t+t_{ij}}) \in A_4, k \in L \tag{3.13}$$

$$d^r_{i_t} \geq 0 \quad \forall i_t \in V \tag{3.14}$$

$$s^r_{i_t} \geq 0 \quad \forall i_t \in V \tag{3.15}$$

$$r_{i_t} \geq 0 \quad \forall i_t \in V \tag{3.16}$$

$$\bar{r}_{i_t} \geq 0 \quad \forall i_t \in V \tag{3.17}$$

4. Genomförande: Utveckling av modeller för dynamisk prissättning

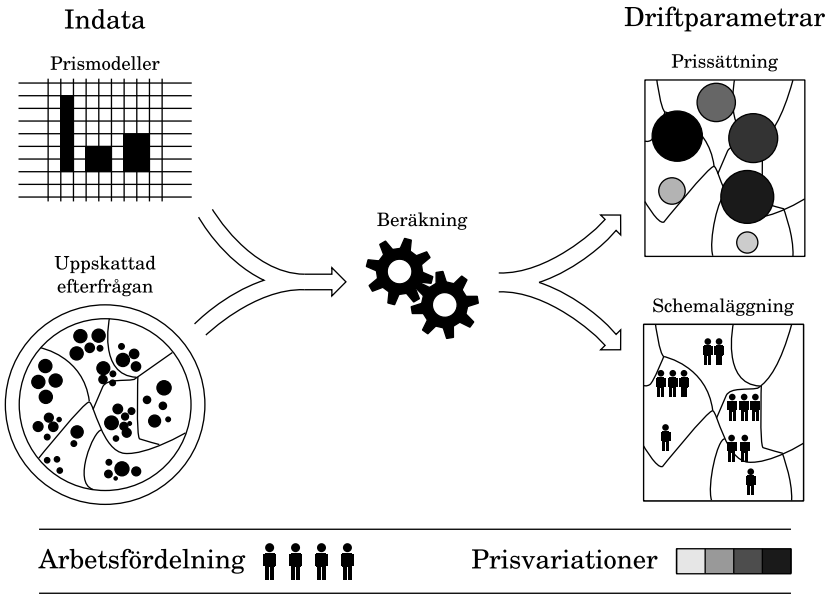
Detta kapitel redogör för utvecklingen av de modeller för driftoptimering genom dynamisk prissättning som legat till grund för implementationen av mjukvaruprototypen.

4.1. En enkel metamodell av dynamisk prissättning

För att kunna utveckla modeller för dynamisk prissättning på ett strukturerat sätt gjordes följande nedbrytning i komponenter, som kunde utvecklas separat:

- **Demand prediction** modellerar hur efterfrågan på bilar i tid och rum förutspås.
- **Prismodell** säger på vilket sätt, hur mycket, och för vad kunden ska betala.
- **Beräkningsmodell** är matematiska beskrivningar av elbilspooler som används för att beräkna priser dynamiskt.

Beräkningsmodellen innehåller en mängd faktorer som antas påverka bilpoolens kostnadseffektivitet, varav ett exempel kan vara schemaläggning av anställda. Figur 4.1 illustrerar hur beräkningsmodellerna behandlar indata för att ta fram driftparametrar till bilpoolen.



Figur 4.1. Enkel metamodell av driftoptimering genom dynamisk prissättning.

Prismodellen är särskilt viktig att beakta i ett inledande skede, då den sätter gränserna för vad som är möjligt att uppnå med dynamisk prissättning. Det första steget i utvecklingen utgjordes därför av utformandet av en lämplig prismodell.

4.2. Prismodeller

En *prismodell* definieras här som en beskrivning av hur man tar betalt för en tjänst eller vara. För bilpooler är ett exempel på en prismodell en fast startavgift, en fast minutavgift och en platsberoende avlämningsrabatt. Mer komplexa prismodeller skulle även kunna inkludera maximal timkostnad eller kostnader som beror på körsträckans längd.

Detta avsnitt behandlar de resonemang som fördes under utvecklandet av den prismodell som användes i de två modellerna för dynamisk prissättning som beskrivs i avsnitten 4.3 och 4.4. De aspekter av prismodeller som undersöktes under utvecklingen var:

- Faktorer som kan leda till missbruk
- Geografiska prisindelningar
- Styrförmåga

Styrförmåga syftar på till vilken grad en prismodell möjliggör påverkan på en användares beteende medelst dynamisk prissättning. Exempelvis har en prismodell som bara tillåter rabatter sämre styrförmåga än en modell som tillåter både rabatter och avgifter.

4.2.1. Faktorer som kan leda till missbruk

En viktigt princip som användbara kombinationer av prismodell och prissättningsalgoritm bör följa är att enbart belöna användare för handlingar som gynnar systemet. Speciellt bör det inte vara möjligt för användare att missbruka systemet, antingen för egen vinning eller i syfte att störa bilpoolens drift.

Vid utformning av prismodeller lämpade för friflytande bilpooler har en huvudsaklig typ av potentiellt missbruk påvisats: Fall där bokningar och resor görs i korta cykler, och start- och slutposition alltså är densamma. Ett exempel på sådant missbruksscenario beskrivs nedan.

Antag en prismodell med ett fast minutpris, rabatt vid avlämning av bilar samt bokningskostnad vid upphämtning. Antag också att för en viss bil har bokningskostnaden satts till 10 kr, och att en bilavlämning i området kring bilen ger en rabatt på 15 kr. Antag slutligen att minutpriset är 2 kr. En kund kan i denna situation boka bilen, för att sedan omedelbart avsluta bokningen. Kundens totala kostnad blir $10+2\,\text{kr} = 12\,\text{kr}$, men samtidigt erhåller kunden 15 kr i rabatt och gör alltså en vinst på 3 kr per bokning.

Om inte speciella begränsningar införs kan bokningar och avlämningar ske i mycket snabb följd. Om en kund kan automatisera denna process kan vederbörande snabbt göra stora vinster på bilpoolens bekostnad. Modeller där sådana sårbarheter påvisats förkastades eller bearbetades vidare för att åtgärda problemen.

4.2.2. Geografiska prisindelningar

En geografisk prisindelning utgörs av avgränsade områden, där varje område associeras med en eller flera kostnader. En friflytande bilpool skulle kunna använda en sådan prisindelning för att sätta upphämtnings- och avlämningskostnader.

Prismodeller som sätter priser efter en zonindelning får vissa problem om zonerna är för stora. Med stora zoner måste stora variationer i pris mellan angränsande zoner tillåtas: Om prisskillnaden mellan närliggande zoner begränsas, begränsas även de totala prisskillnaderna inom driftområdet (då driftområdet är indelad i ett ändligt antal zoner). Om stora prisskillnader i sin tur inte tillåts inom driftområdet förlorar den dynamiska prissättningen sin styrförmåga, då inga starka ekonomiska skäl kan föreligga för att en kund ska välja en viss bil eller avlämningsplats före en annan.

Stora prisskillnader mellan angränsande zoner är inte önskvärt då obetydliga skillnader i placeringen av en bil nära zongränser kan ge markanta skillnader i pris. Avgiften eller rabatten för en upphämtning eller avlämning bör reflektera dess påverkan på bilpoolens balans (*nytta*). När prisincitament varierar mycket över korta avstånd reflekterar de inte nyttan, då skillnaden i nytta mellan att lämna av en bil på två närliggande punkter är obetydlig.

4.2.3. Den symmetriska prismodellen

Den prismodell som i slutändan valdes, var en med fast minutpris och variabla upphämtnings- och avlämningskostnader (eller rabatter). Prismodellen benämns *symmetrisk*, då sambandet (avlämingsrabatt = upphämtningskostnad) gäller överallt i driftområdet. Denna prismodell användes i båda de utvecklade modellerna för dynamisk prissättning.

4.3. Modell 1: Kek-baserad dynamisk prissättning

Den första modellen för dynamisk prissättning som utarbetades byggde på att försöka utvidga Kek-modellen till det friflytande bilpoolskonceptet, och att integrera dynamisk prissättning i denna.

4.3.1. Demand prediction

Den modell för demand prediction som utarbetades baserades på historisk användningsdata. Detta motiverades av att åtminstone en tidigare känd studie, i form av Kek et al. (2009), använt modeller som byggt på samma princip. Modellen beskrivs i avsnitt 6.1.

4.3.2. Beräkningsmodell

Beräkningsmodellen som utarbetades utgick från Alvina Keks optimeringsmodell för schemaläggning av anställda i bilpooler med envägsresor. Modellen valdes för att dess effektivitet verifierats (Kek et al., 2009, s. 128), och för att den ansågs relativt enkel att implementera. Den bedömdes även kunna utvidgas till att inkludera dynamisk prissättning. Tanken var att sedan anpassa denna stationsbaserade modell till att användas för dynamisk prissättning i friflytande bilpooler.

Införseln av dynamisk prissättning skedde iterativt under det att idéer formulerades, utforskades och granskades. Idéerna konvergerade slutligen till en matematisk formulering som var tillsynes befriad från logiska fel och uppenbara prestandaproblem. I den slutliga formuleringen, som återfinns i avsnitt 6.1, gjordes ett försök att modellera tillgång och efterfrågan, och effekterna prisjusteringar har på dessa. Under utvidgningens gång upptäcktes emellertid ett antal egenskaper i den omodifierade Kek-modellen som gjorde den olämplig att utgå ifrån. Dessa egenskaper härstammade från både själva modellen (se nedan) och dess implementering (se avsnitt 5.3.1).

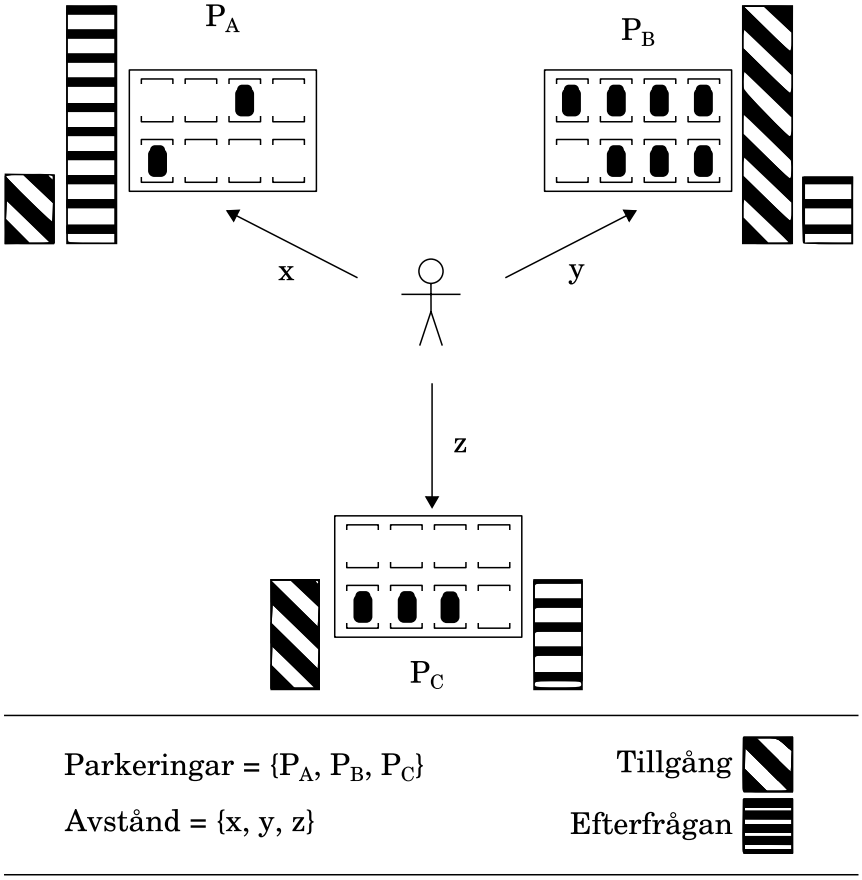
Svårigheter med integration av dynamisk prissättning

Ett problem med utvidgningen av Keks modell till att inkludera dynamisk prissättning är att den ursprungliga modellen inte tar hänsyn till stationernas geografiska positioner. Därmed finns det inget enkelt sätt att modellera en kund som väljer mellan närliggande stationer. Figur 4.2 visar varför geografiska avstånd påverkar kundens val. Givet att avstånden till de lediga bilarna – x, y och z – är ungefär lika långa kommer kunden att kunna göra sitt val baserat på upphämtningskostnaden. Denna kostnad kan dock antas ha mindre inverkan på detta val om skillnaderna i gångavstånd är betydande.

4.4. Modell 2: Fordonsbalans-baserad dynamisk prissättning

Eftersom Kek-modellen är stationsbaserad kunde driftoptimeringen inte direkt tillämpas på friflytande bilpooler. Försök gjordes att approximera den stationlösa egenskapen med ett stort antal stationer (detta beskrivs i avsnitt 5.3.1), dock med begränsad framgång. Att modellen inte representerar stationer spatialt utgjorde en ytterligare svårighet, liksom att den implementerade målfunktionen blev icke-konvex (se avsnitt 6.1).

För att åtgärda dessa problem togs en ny modell fram. Denna nya modell baseras på skillnad mellan tillgång och efterfrågan i fordonsfördelningen. Genom att studera denna skillnad ges möjlighet att analysera huruvida, och till vilken grad, behovet av bilar är uppfyllt på olika platser inom driftområdet. En sådan studie kan ligga till grund för beslut om relokeringar och dynamisk prissättning. Beslut om relokeringar kan tas om stora avvikelser från balansen upptäcks och det inte finns skäl att tro att systemet kommer att balanseras spontant.



Figur 4.2. Kundens valmöjligheter vid hämtning av bilar. Kundens val antas bero på närliggande parkeringars upphämtningskostnad och avståndet till dem.

Då denna modell inte är stationsbaserad beräknas istället ett balansvärde för varje punkt i driftområdet. Dessa balansvärden kan visualiseras i form av en värmekarta, vilket beskrivs i avsnitt 7.3.1. Visualiseringen kan användas i en allmän bedömning av bilpoolens balans för att exempelvis upptäcka icke önskvärda trender i bilfördelningen. På så sätt kan effektiviteten hos olika strategier för dynamisk prissättning utvärderas, då man med modellen kan avgöra om och var eventuell obalans uppstår.

4.4.1. Demand prediction

Två principer har följts för att utveckla metoder för beräkning av efterfrågan i denna modell. Dessa två principer utgjordes av: Enkel beskrivning och implementation, samt interpolering mellan punkter som antagits kunna tillskrivas ett visst värde på efterfrågan.

Två modeller implementerades, en baserad på tidigare gjorde bokningar (se avsnitt 6.2.1) och en baserad på speciella platser där efterfrågan antas vara känd (se avsnitt 6.2.2).

4.4.2. Beräkningsmodell

Ett antal olika metoder för beräkning av balans har utforskats, men alla har varit baserade på samma grundläggande idé: Avvikelsen från balans i en punkt kan beskrivas som efterfrågan på bilar i punkten subtraherad från tillgången på bilar i samma punkt. En punkt där värdet på denna avvikelse är lika med 0 är alltså balanserad.

Olika metoder för själva beräkningen av balans har också implementerats och utvärderats. Även dessa har varit variationer på samma tema. Alla metoder som utforskats bygger på att tillgången på bilar beräknas för varje punkt i driftområdet genom interpolation mellan de faktiska bilarnas positioner.

En detaljerad beskrivning av den implementerade algoritmen för balansberäkning återfinns i avsnitt 6.2.

Prisberäkning

Utgångspunkten för beräkningen av priser i denna modell är balansavvikelser. Positiva avvikelser (överskott på bilar) ska ge upphämtningsrabatter och avlämningsavgifter. Det motsatta ska gälla för negativa avvikelser.

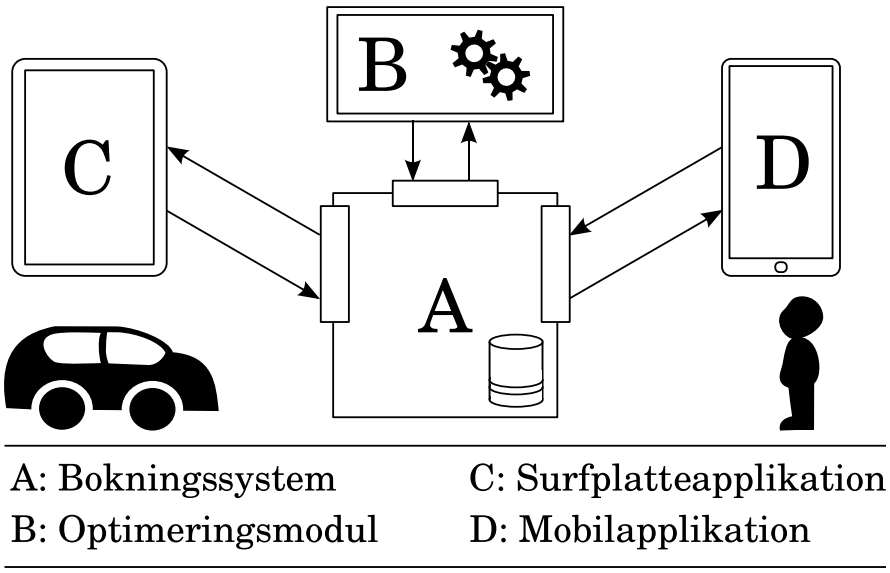
De rena avvikelsevärdena har dock inte nödvändigtvis rätt magnitud för att vara lämpliga som priser. En funktion som omvandlar dessa avvikelsevärden till priser och rabatter behövs. Enklaste möjliga funktion för detta implementerades.

5. Genomförande: Utveckling av mjukvaruprototyp

I detta kapitel beskrivs implementationen av den mjukvaruprototyp vari utvecklade teoretiska modeller integrerades. Avsnittet berör designval som styrde utvecklingen, samt intressanta problem som uppkom vid implementationsarbetet.

5.1. Delsystem

Fyra delsystem utvecklades: ett bokningssystem, en optimeringsmodul, en mobilapplikation och en surfplatteapplikation. Deras respektive syften och utformning beskrivs nedan. En översiktlig bild av systemets utformning ges i figur 5.1.



Figur 5.1. Kopplingar mellan IT-lösningens olika delsystem. Pilarna representerar kommunikation över HTTP.

- **Bokningssystemet** utformades som en central punkt för lagring och hämtning av bilrelaterad data. Detta kompletterades med nödvändig affärslogik för sökningar, bokningar och hantering av kunddata.

- **Optimeringsmodulen** utvecklades för att samla optimeringsrelaterad funktionalitet på ett ställe. Separationen av bokningssystemet och optimeringsmodulen medförde möjligheten att delegera optimeringsmodulens krävande beräkningar till andra datorer, på geografiskt åtskilda platser.
- **Mobilapplikationen** utvecklades så att bilpoolens kunder skulle kunna hitta och boka närliggande bilar. Implementeringen av funktionaliteten i just en applikation för smarttelefoner motiverades av möjligheten att utnyttja GPS-data för att till exempel hitta den närmaste bilen.
- **Surfplatteapplikationen** utvecklades för att köras i surfplattor, monterade i bilpoolens bilar. Dess syfte var att låta kunder rapportera bilstatus vid början av resor, samt ta del av prisinformation under resor. Utöver det planerades den regelbundet rapportera bilens position och laddningsstatus till bokningssystemet, för att kunna använda datan i optimeringsmodeller. Prisvisualiseringen kan anses vara dess viktigaste funktion, då kunder måste göras medvetna om prisskillnader för att dynamisk prissättning ska kunna ha en styrande effekt.

5.2. Plattformar och programspråk

I detta avsnitt redogörs för valen av de plattformar, programspråk och kommunikationsprotokoll som användes vid utvecklingen av mjukvaran.

5.2.1. Android

Vid beslutet om utvecklingsplattform för mobil- och surfplatteapplikation övervägdes både en applikation utvecklad direkt mot det mobila operativsystemet, samt en plattformsoberoende *Rich Internet application*. Då det planerade gränssnittet och funktionaliteten var beroende av användarposition uteslöts webbapplikationer då åtkomst till systemfunktioner vanligtvis är begränsad. Sådan åtkomst är nödvändig för att erhålla exempelvis positionsdata från inbyggda GPS-mottagare.

Av de två vanligaste operativsystemen för smarttelefoner, Apples iOS och Googles Android, valdes Android på grund den fria och lättillgängliga utvecklingsmiljön. Valet styrktes av att endast Android-baserade telefoner fanns att tillgå för att testa applikationerna under utvecklingen.

Genom att välja Android inskränktes den potentiella användarkretsen till individer med kompatibla enheter. En alternativ lösning hade varit att bygga en mobilapplikation baserad på HTML 5 och Javascript med PhoneGap[1] eller annat liknande verktyg. Eftersom applikationen i ett första skede inte skulle släppas till verkliga användare bedömdes detta var mindre intressant.

[1] Ett verktyg för utveckling av plattformsoberoende mobilapplikationer, se `http://phonegap.com` för mer information.

5.2.2. Java

Det beslutades i ett tidigt skede att Java skulle användas som primärt utvecklingsspråk. Beslutet underbyggdes av valet av mobil plattform, då Java är det primära utvecklingsspråket för Android. Alla större mjukvarukomponenter i IT-systemet utvecklades därför i Java.

5.2.3. RDBMS/PostgreSQL

Bokningssystemets uppgifter består till stor del av att hämta och lagra data. Därför beslutades det att ett existerande DBMS (Database Management System) skulle användas, då den typen av mjukvara är svår att utveckla och väl fungerande, kostnadsfria och öppna lösningar finns.

På grund av tidigare erfarenhet inom gruppen samt behovet att kunna göra komplicerade sökningar i lagrad data, beslutades det att ett RDBMS (Relational Database Management System) skulle användas. Valet föll på PostgreSQL på grund av dess inbyggda stöd för lagring och manipulation av, samt förfrågningar baserade på, geografisk data.

5.2.4. HTTP

Allt datautbyte och alla funktionsanrop mellan delsystem sker över HTTP. Under utvecklingen fanns ingen fast uppsättning krav och därmed inte heller någon specifikation av dataformat. Som följd valdes HTTP framför mer specialiserade protokoll på grund av behovet av att kunna sända godtycklig data till ett potentiellt stort antal olika klienter och möjliggöra enkel utveckling av nya klientapplikationer.

Då standardiserade bibliotek, och i vissa fall inbyggt stöd, för HTTP existerar för många programspråk och plattformar[2] kan utveckling av klientenheter ske utan krav på stöd för proprietära tekniska lösningar. Detta öppnar upp möjligheter för tredjepartsutvecklare att arbeta mot systemet, vilket i sin tur främjar utbudet av applikationer för slutanvändare.

Åtkomsten till bokningssystemets funktioner och data begränsas genom HTTP Basic-autentisering. Denna metod för åtkomstkontroll skickar användarnamn och lösen i base64-kodning vilket ur säkerhetssynpunkt är likvärdigt med klartext (Internet Engineering Task Force, 1999). Detta problem kan lösas genom att använda krypterad överföring i form av HTTPS (Hypertext Transfer Protocol Secure). Ingen kryptering används i det implementerade systemet och det kan alltså inte anses vara säkert.

[2] libcurl har bindningar till 40 olika språk (Stenberg, 2012).

5.2.5. Jersey/Grizzly

Som referensimplementationen av JAX-RS (The Java API for RESTful Web Services) är Jersey ett ramverk för utveckling av REST (Representational State Transfer)-baserade webbtjänster (Oracle Corporation, 2012). Jersey använder sig av webbservern Grizzly.

5.3. Implementation av optimeringsmodulen och Kek-baserad dynamisk prissättning

Optimeringsmodulen skapades för att kunna integrera den Kek-baserade dynamiska prissättningen i mjukvaruprototypen på ett modulärt sätt.

Som grund vid implementationen av Keks modell användes trestegsprocessen OTS (Optimization Trend Simulation) som beskrivs i Kek et al. (2009). Detta ledde till den strukturmässiga utformningen av optimeringsmodulen (se avsnitt 7.4). I optimeringsmodulen implementerades sedan modellen för dynamisk prissättning beskriven i avsnitt 4.3.

5.3.1. Implementation av dynamisk prissättning

Den Kek-baserade modellen för dynamisk prissättning implementerades i två matematiska programmeringsspråk, OPL (Optimization Programming Language) och GMPL (GNU Mathematical Programming Language).

Implementationen i GMPL ledde till upptäckten att modellen var kvadratisk, då lösaren `glpsol` (som användes till att lösa GMPL-koden) inte kan hantera kvadratiska problem (MIQP-problem) (Makhorin, 2010). På grund av detta överfördes hela modellen till OPL-kod, och IBM ILOG CPLEX Optimization Studio användes som lösare.

För att kunna lösa kvadratiska problem kräver CPLEX att den kvadratiska formen i målfunktionen är positivt definit, och därmed också konvex. Målfunktionen förlorar konvexitetsegenskapen när antalet stationer och tidssteg blir för stort. Det finns lösare som inte ställer dylika krav på konvexitet,[3] men projektgruppen hade inte tillgång till sådana verktyg.

Anpassning till friflytande bilpooler och analys av tidskomplexitet

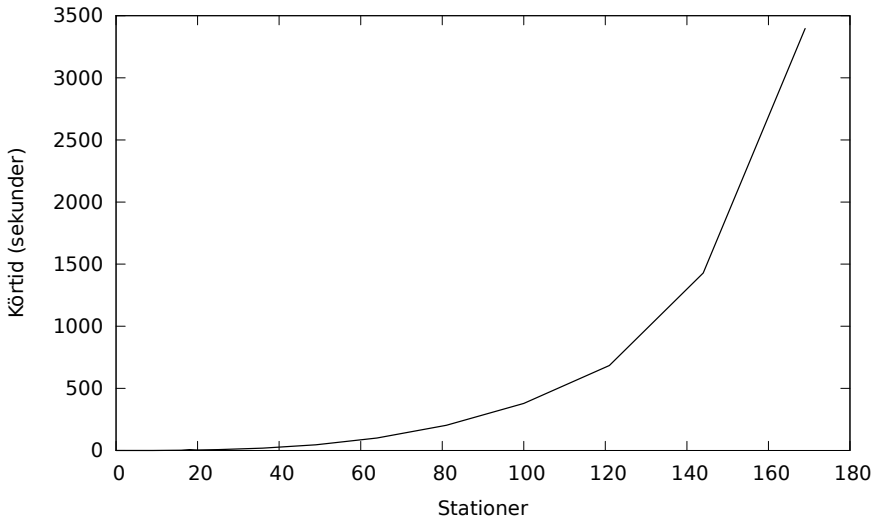
Eftersom Keks modell är utvecklad med en stationsbaserad bilpool i åtanke krävdes en översättning till friflytande bilpooler för att modellen skulle bli användbar. Denna översättning kan göras genom att i Keks modell placera ut ett stort antal virtuella stationer och låta varje station representera ett geografiskt område. Dessa områden skulle tillsammans utgöra en uppdelning av driftområdet och på så sätt skulle den geografiska

[3] BARON Global Optimization Software är ett exempel på en sådan lösare.

prisuppdelningen kunna implementeras. Denna insikt föranledde en komplexitetsanalys av algoritmen med avseende på antal stationer, i syfte att undersöka huruvida översättningen skulle fungera i praktiken. En fullständig redogörelse för komplexitetsanalysen återfinns i bilaga A.

Analysen gjordes med avseende på antalet ingående stationer för att bilda en uppfattning om hur en mycket fin indelning av zoner skulle te sig för Kek-modellen i fallet där stationer fick representera zoner.

Figur 5.2 visar tidskomplexiteten i antal stationer, för 4 tidssteg. Andra värden för tidsstegen gav liknande resultat (och alltid samma komplexitetsmönster). Algoritmen uppvisade exponentiell tidskomplexitet i asymptotiskt hänseende, vilket kan ses i figur 5.2. Då antalet stationer skulle behöva vara mycket stort ($n \gg 1000$) är det uppenbart att denna lösning inte kan beräknas inom rimliga tidsgränser.



Figur 5.2. Beräkningstid som funktion av antal stationer, för ett konstant antal (4) tidssteg.

5.4. Utveckling av bokningssystemet

Bokningssystemet utvecklades till en början parallellt med optimeringsmodulen. Detta gjordes bland annat för att låta bokningssystemet tillhandahålla information om exempelvis restider och relokeringskostnader åt optimeringsmodulen. Vid ett senare skede utökades gränssnittet mot optimeringsmodulen och ett nytt gränssnitt byggdes för mobilapplikationen. Slutligen integrerades den fordonsbalans-baserade dynamiska prissättningen i bokningssystemet, och ett gränssnitt skapades för surfplatteapplikationen.

5.4.1. Implementation av balans-baserad dynamisk prissättning

Den fordonsbalans-baserade prissättningen implementerades direkt i bokningssystemet snarare än i optimeringsmodulen. Då priserna måste uppdateras vid varje bokning och avbokning, och optimeringsmodulen byggdes för att själv initiera prissättningen, skulle den ha behövt en egen server för att lyssna efter bokningssystemets förfrågningar vid varje tillståndsförändring.

Inledningsvis lagrades zonindelningen i databasen, med zoner som anpassats för den Kek-baserade dynamiska prissättningen. Explicita pristabeller skapades också för att de priser som genererats av optimeringsmodulen skulle kunna användas. Vid skiftet till den fordonsbalans-baserade modellen, flyttades zonindelningen till interna datastrukturer i bokningssystemet. De explicita pristabellerna ersattes då av en implicit tabell beräknad från tabeller över geografisk fördelning av tillgång och efterfrågan på bilar.

5.5. Utveckling av Android-applikationer

Under utvecklingen av Android-applikationerna togs särskild hänsyn till den relativt begränsade batterikapacitet och beräkningsförmåga som i allmänhet återfinns hos mobila enheter. De krävande beräkningar som krävs för att rendera värmekartor med prisinformation delegerades därför till bokningssystemet.

Då applikationernas funktionalitet till stor del är beroende av informationsutbyte över Internet, och dataöverföring i regel medför en hög batterikonsumtion, genomfördes åtgärder för att hålla sådan överföring till ett minimum. Tillståndsberoende data, till exempel prisinformation, sparas lokalt och uppdateringar utförs endast om tillståndet har förändrats. Speciellt de värmekartor som presenterar prisinformation i den bilmonterade surfplattan har förhållandevis stor filstorlek, vilket gör det viktigt att minimera antalet gånger de måste skickas över nätverket.

6. Resultat: Utvecklade modeller för dynamisk prissättning

I detta kapitel beskrivs de modeller som utvecklades för applicering av dynamisk prissättning i friflytande bilpooler. Den i slutändan övergivna Kek-modellen och den implementerade balansmodellen redogörs för i varsitt avsnitt. För implementationsdetaljer, se kapitel 7.

6.1. Modell 1: Kek-baserad dynamisk prissättning

Modellen beskriven i detta avsnitt är en variant av Kek-modellen som är modifierad för att tillåta variabla priser för upphämtning och avlämning vid stationerna.

6.1.1. Demand prediction

Modellen för demand prediction bygger på den modell som användes i Kek (2006). På samma sätt som Kek, användes speciella tidsintervall i återkommande perioder till att förutspå framtida efterfrågan (exempelvis kl. 13:00–14:00 på måndagar). Framtida efterfrågan för varje station, under dessa tidsintervall, approximerades genom medelvärdet av efterfrågan i samma tidsintervall från tidigare veckor. Andra tidsperioder än veckor skulle också kunna användas. Utöver detta gjordes även en modifikation av modellen som Kek använde, där data hämtad från tidigare datum viktades till att ha mindre betydelse än nyare data.

6.1.2. Beräkningsmodell

Principen bakom den utvecklade modellen bygger på att i målfunktionen för Keks modell lägga till tids- och stationsbundna variabler för att representera rabatter och varierbar efterfrågan, i ett försök att sammanföra relokeringskostnader med variabla priser.

Redogörelsen för hur Kek-modellen har modifierats görs här enbart med hänsyn till variabla priser för upphämtning av bilar. Införandet av variabla avlämningspriser gjordes helt analogt.

För upphämtningskostnaderna innebar modifikationen av målfunktionen att två kostnader lades till. Den första kostnaden avser de vinster/förluster bilpoolen gör i samband med att rabatter/avgifter ges till kunderna. Den andra kostnaden avser de vinster/förluster bilpoolen gör i samband med att fler/färre kunder nyttjar bilpoolen till följd av prisförändringarna.

Dessa två kostnader utgör således en avvägning mellan höga priser och hög efterfrågan, och modellerar hur varierad prissättning kan användas för att styra marknaden på ett gynnsamt sätt. På platser där lägre efterfrågan är önskvärt, kan priserna höjas, och vice versa. Den andra kostnaden använder sig av antaganden om hur kunder kommer att påverkas av den dynamiska prissättningen, beskrivna av den priselasticitetsfunktion som finns formulerad i bivillkor (6.15).

Modifierad målfunktion

De extra kostnaderna som införts medför att tre nya termer införs i målfunktionen. Låt K vara den ursprungliga målfunktionen i Kek-modellen, som beskriven i (3.1). Den modifierade målfunktionen är då summan av K och de tre nya termerna (6.1).

$$\begin{aligned} K - \sum_{i_t \in V} rb_{i_t} d^r_{i_t} + \sum_{i_t \in V} rb_{i_t} nb_{i_t} + \\ + \sum_{i_t \in V} \left(\operatorname{sgn}(vd_{i_t}) \cdot (c_d - w) \cdot \min\left(|vd_{i_t}|, d^r_{i_t} \right) - vd_{i_t}(c_d - w) \right) \end{aligned} \tag{6.1}$$

$$\text{där } w = \begin{cases} \dfrac{rb_{i_t}}{2} & \text{om } vd_{i_t} < 0 \\ rb_{i_t} & \text{om } vd_{i_t} \geq 0 \end{cases}$$

Den tredje termen i (6.1) avser inkomster/kostnader bilpoolen får som följd av rabatter/avgifter, och den fjärde avser inkomster/kostnader orsakade av förändringar i efterfrågan som följd av dynamisk prissättning. Den andra termen är en justering av kostnaden att avvisa en kund (c_d), som tar hänsyn till prisjusteringen.

För att kunna representera prisjusteringarna och förändringar i efterfrågan i målfunktionen infördes en ny beslutsvariabel (rb_{i_t}) och två nya beroende variabler (vd_{i_t} och nb_{i_t}).

- rb_{i_t} Beslutsvariabel som anger den rabatt som en kund får för upphämtning av en bil vid en tidpunkt t och station i
- vd_{i_t} Beroende variabel som anger hur stor ökning av efterfrågan station i får för upphämtning vid tidpunkten t som följd av prisförändringar. Efterfrågan ändras proportionerligt mot resors priselasticitet.
- nb_{i_t} Beroende variabel som anger hur många bilar som hämtas vid station i under tidssteg t

Dessa variabler är inte begränsade att anta positiva värden, och en negativ rabatt kan således tolkas som en kostnad för kunden (och en vinst för bilpoolen). Likaså kan ett negativt värde på vd_{i_t} tolkas som förlorade kunder vid den stationen och tiden.

Den fjärde termen i (6.1) gör modellen kvadratisk som följd av multiplikationen av vd_{i_t} med w, och därmed indirekt med beslutsvariabeln rb_{i_t}. Därmed görs problemet till ett MIQP-problem.

Justering av kostnad för kundavvisande (term 2)

Konstanten som anger kostnaden för att avvisa en kund, c_d, ersattes med det variabla uttrycket $c_d - rb_{i_t}$. Detta då större förluster görs vid avvisning av kunder då priserna är höga, och det motsatta för låga priser[1]. Detta innebar att kostnaden

$$c_d \sum_{i_t \in V} d^r_{i_t}$$

från målfunktion (3.1) skulle kunna ändras till (6.2) för att införa denna ändring.

$$\sum_{i_t \in V} d^r_{i_t}(c_d - rb_{i_t}) \tag{6.2}$$

Istället lades följande term till i målfunktionen för att behålla K oförändrad.

$$\sum_{i_t \in V} -rb_{i_t} d^r_{i_t} \tag{6.3}$$

Kostnaden att avvisa en kund antogs här öka/minska med samma belopp som prisjusteringarna, varför rb_{i_t} återanvändes.

Extra kostnader som följd av prissänkningar (term 3)

Nedan beskrivs den tredje termen i målfunktionen i form av kostnader. Observera att denna term kan vara negativ som följd av prishöjningar, vilket skulle innebära att termen istället skulle representera intäkter.

Prissänkningar leder till att bilpoolen förlorar intäkter från kunder som potentiellt hade betalat högre priser[1]. Detta modelleras genom att i målfunktionen lägga till en kostnad:

$$\sum_{i_t \in V} rb_{i_t} nb_{i_t} \tag{6.4}$$

[1]Uteblivna intäkter räknas som kostnader eftersom Keks modell enbart modellerar kostnader.

Antalet bilar kunderna tar vid varje tidssteg och station multipliceras här med rabatterna dessa får för att ta bilarna. Detta utgör alltså de direkta kostnader rabatterna åsamkar bilpoolen.

Variabeln nb_{i_t} kan beräknas genom efterfrågan ($vd_{i_t} + d_{i_t}$) - nekad efterfrågan ($d^r_{i_t}$): $nb_{i_t} = vd_{i_t} + d_{i_t} - d^r_{i_t}$.

Extra kostnader som följd av sänkt efterfrågan (term 4)

Detta avsnitt beskriver och härleder den fjärde termen i målfunktionen i form av kostnader. Denna term kan även vara negativ, som följd av höjd efterfrågan, vilket skulle innebära att termen istället representerade intäkter.

De kostnader bilpoolen får som följd av sänkt efterfrågan ges av (6.5). Låt $\widetilde{d}^r_{i_t}$ vara det totala antalet avvisade kunder då $vd_{i_t} = 0$.

$$\sum_{i_t \in V} \left(\text{sgn}(vd_{i_t}) \cdot (c_d - w) \cdot \min\left(|vd_{i_t}|, \widetilde{d}^r_{i_t} \right) - vd_{i_t}(c_d - w) \right) \tag{6.5}$$

$$\text{där } w = \begin{cases} \dfrac{rb_{i_t}}{2} & \text{om } vd_{i_t} < 0 \\ rb_{i_t} & \text{om } vd_{i_t} \geq 0 \end{cases}$$

I fallet $vd_{i_t} < 0$ (efterfrågan förloras på grund av höjda priser), representerar kostnaderna i (6.5) förlusten av de eventuella intäkter bilpoolen annars skulle fått av den efterfrågan som nu förlorades. När $vd_{i_t} \geq 0$ (ökad efterfrågan), representerar (6.5) istället de intäkter bilpoolen får som följd av extra uthyrningar.

De bägge fallen $vd_{i_t} \geq 0$ och $vd_{i_t} < 0$ måste behandlas separat då dessa måste ta hänsyn till olika kostnader.

Fallet $vd_{i_t} < 0$ (sänkt efterfrågan) I fallet då $vd_{i_t} < 0$, har term 4 följande form:

$$\sum_{i_t \in V} \left(- \left[c_d - \frac{rb_{i_t}}{2} \right] \cdot \min\left(|vd_i t|, \widetilde{d}^r_{i_t} \right) - vd_{i_t} \left[c_d - \frac{rb_{i_t}}{2} \right] \right) \tag{6.6}$$

Den totala kostnaden av de eventuella intäkter bilpoolen annars skulle fått av den efterfrågan som nu förlorades, utgörs alltså av två termer. (6.7) representerar en intäkt, och (6.8) representerar en kostnad.

$$- \left(c_d - \frac{rb_{i_t}}{2} \right) \cdot \min\left(|vd_{i_t}|, \widetilde{d}^r_{i_t} \right) \tag{6.7}$$

$$vd_{i_t} \left(c_d - \frac{rb_{i_t}}{2} \right) \tag{6.8}$$

Följande exempel används nedan till att förklara de termer som ingår i (6.6). Exemplet kan även användas till att visa hur en del av den totala kostnaden (6.6) kan representeras som en intäkt (6.7).

Antag att bilpoolen vid en given station och tidpunkt, och utan prishöjningar, hade avvisat en kund en gång att hyra en bil. Antag sedan samma situation, fast där att priserna höjts. Antag att totalt en kund ville avstå från att nyttja bilpoolen i och med prishöjningen (sänkt efterfrågan = 1). I bägge dessa situationer, hade en kund inte kunnat nyttja bilpoolen vid ett tillfälle. Bilpoolen skulle således inte få intäkter från detta tillfälle vid något av fallen. I detta fall hade alltså inte bilpoolen åsamkats några kostnader, trots att efterfrågan sänkts. Om efterfrågan ej avvisats i det första fallet, hade istället bilpoolen gått med förlust på grund av den sänkta efterfrågan.

Detta exempel visar att kostnader från sänkt efterfrågan endast uppstår om efterfrågan kan mötas i fallet där efterfrågan inte sänkts. Det visar även att den sänkta efterfrågan kan representeras som en intäkt, i och med att kunder (efter sänkt efterfrågan) i vissa fall inte längre behöver avvisas.

I (6.6), representerar uttrycket (6.7) den kostnaden bilpoolen skulle ha fått om efterfrågan aldrig sänkts, om (6.10) förfrågningar hade nekats. (6.9) i uttrycket (6.8) representerar den kostnad bilpoolen får för en förlorad efterfrågan, i fallet när obemött efterfrågan bortses ifrån. Detta multipliceras med vd_{i_t} för att få den totala kostnaden för alla efterfrågan som förlorats, utan hänsyn till om dessa efterfrågan möts.

$$c_d - \frac{rb_{i_t}}{2} \tag{6.9}$$

$$\min\left(|vd_{i_t}|, \widetilde{d}^r_{i_t}\right) \tag{6.10}$$

Multiplikationen av faktorerna (6.9) och (6.10) i (6.7) kommer av att antalet kunder som avvisats utan prisförändringar (6.10) (av de efterfrågan som förlorats) multipliceras med kostnaden att avvisa dessa efterfrågan (6.9).

I (6.10) inräknas enbart antalet avvisningar som är relevant för kostnaden som följd av sänkt efterfrågan. Detta innebär att det största antalet avvisningar som kan räknas in, är antalet efterfrågan som minskats. Detta är varför det minsta talet av $|vd_{i_t}|$ och $\widetilde{d}^r_{i_t}$ används.

$\widetilde{d}^r_{i_t}$ härleds enligt följande resonemang till $d^r_{i_t}$, som används i målfunktionen. När kostnader som följd av förlorade intäkter från uthyrningar beräknas, görs detta genom att jämföra bilpoolens tillstånd då vd_{i_t} är i sitt aktuella tillstånd, och hur kostnader hade beräknats om $vd_{i_t} = 0$. Bilpoolens tillstånd utöver beräkningarna för just i_t behöver alltså inte tas hänsyn till. Vi jämför därför enbart skillnaden i kostnader då $\widetilde{d}^r_{i_t}$ används till att beräkna kostnaderna, och då $d^r_{i_t}$ (se avsnitt 3.3.2) används till detta, för ett givet tillstånd i bilpoolen (för några givna värden på beslutsvariablerna). Då tillståndet är detsamma

för båda fallen som jämförs (alla beslutsvariabler är likadana), kommer beslutsvariablerna $\widetilde{d}^r_{i_t}$ och $d^r_{i_t}$ i detta fall vara detsamma. $\widetilde{d}^r_{i_t}$ kan alltså ersättas med $d^r_{i_t}$.

(6.9) förklaras nedanstående stycke.

c_d kommer av att inkomsten från en uthyrning utan att prisförändringar har antagits vara ungefär lika stort som kostnaden att avvisa en kund. Inkomsten från en uthyrning med prisförändringar har antagits vara lika stort som kostnaden att avvisa en kund med prisförändringar ($c_d - rb_{i_t}$). (6.9) använder sig av förenklingen att den genomsnittliga kostnaden bilpoolen åsamkas per förlorad kund (i fallet när minuenden c_d bortses ifrån), är det aritmetiska medelvärdet av kostnaden utan prisförändringar och kostnaden med prisförändringarna (c_d och $c_d - rb_{i_t}$). Denna förenkling gör antagandet att priselasticitetsfunktionen är linjär, och därmed att ungefär lika många kunder väljer att avstå från att använda bilpoolen för varje enhet ökat pris. De kunder som avstått från att använda bilpoolen på grund av en prishöjning, antas alltså i genomsnitt ha gjort det när priset överstigit medelvärdet av c_d (priset utan prisförändringar) och $c_d - rb_{i_t}$ (priset med prisförändringar). Det aritmetiska medelvärdet ges av:

$$\frac{c_d + c_d - rb_{i_t}}{2} = c_d - \frac{rb_{i_t}}{2}$$

Fallet $vd_{i_t} \geq 0$ (höjd efterfrågan) Även i detta fall måste två termer tas hänsyn till. För det första fås vinster för de efterfrågan som ökats i form av nya kunder som nyttjar bilpoolen. Detta kan beskrivas av (6.11)

$$vd_{i_t}(c_d - rb_{i_t}) \tag{6.11}$$

Den första faktorn i (6.11) representerar (som i fallet $vd_{i_t} < 0$) även här priset för en uthyrning. För det andra fås förluster för de nya efterfrågan som inte kan mötas av bilpoolen (nekade kunder). Detta kan beskrivas av (6.12).

$$(cd - rb_{i_t}) \cdot \min(vd_{i_t}, d^r_{i_t}) \tag{6.12}$$

Den första faktorn i (6.12) är den modifierade avgiften för att neka kunder (som i fallet (6.3)). Den andra faktorn representerar antalet efterfrågan som nekades, av de efterfrågan som tillfördes (analogt med fallet $vd_{i_t} < 0$). Produkten av dessa faktorer ger därmed kostnaden av att neka den tillförda efterfrågan.

Den totala kostnaden bilpoolen får som följd av att $vd_{i_t} \geq 0$ blir alltså kostnaderna (6.12) minus vinsterna i (6.11). Den totala kostnaden ges av (6.13)

$$\sum_{i_t \in V} \big((c_d - rb_{i_t}) \cdot \min(|vd_{i_t}|, d^r_{i_t}) - vd_{i_t}(cd - rb_{i_t})\big) \quad vd_{i_t} \geq 0 \tag{6.13}$$

Kombineras (6.6) och (6.13) fås sedan term 4 i (6.5).

Modifierade bivillkor

Två nya bivillkor introducerades till Kek-modellen, och de existerande bivillkoren modifierades något. De tillagda bivillkoren visas i (6.14) och (6.15).

$$vd_{i_t} + d_{i_t} \geq 0 \quad \forall t \in T, i \in S \tag{6.14}$$

$$rb_{i_t} = pe \cdot vd_{i_t} \quad \forall t \in T, i \in S \tag{6.15}$$

Bivillkor (6.14) säger att efterfrågan (den konstanta + variabla efterfrågan) inte får vara negativ. Bivillkor (6.15) är ett förenklat sätt att modellera priselasticiteten för bokningar. Det uttrycker sambandet mellan pris och efterfrågan på följande sätt: en rabatt *pe* ökar efterfrågan med en kund, och en avgift pe minskar efterfrågan med en (1) kund.

Slutligen gjordes ett antal modifikationer av bivillkoren i Keks modell. Modifikationerna bestod i att ersätta termen d_{i_t} (basefterfrågan) med $d_{i_t} + vd_{i_t}$ (basefterfrågan + variabel efterfrågan) överallt där den förekom bland bivillkoren.

6.2. Modell 2: Fordonsbalans-baserad dynamisk prissättning

Denna modell bygger på att för varje (geografisk) punkt i staden beräkna två värden: önskat antal bilar och faktiskt antal bilar. Skillnaden mellan dessa värden i en punkt representerar då avvikelsen från den önskvärda bilfördelningen. När skillnaden har beräknats kan en lämplig funktion appliceras på den för att beräkna avgifter och rabatter för upphämtning samt avlämning i punkten. Modellen implementerades med en mycket enkel sådan funktion: Skillnaden multipliceras med en konstant och resultatets magnitud begränsas.

6.2.1. Interpolerande algoritm för tillgång och efterfrågan

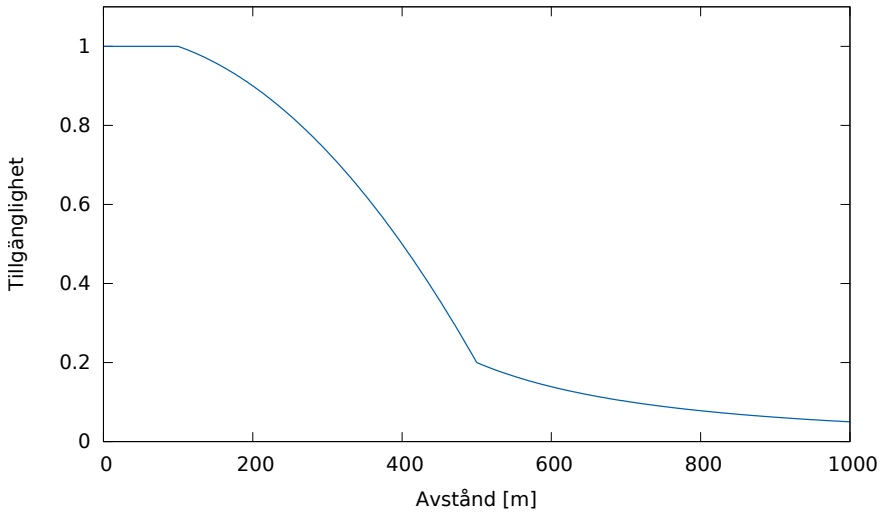
Densiteten av bilar i en punkt beräknas genom att först tilldela varje bil ett värde B_x som beror på avståndet mellan den aktuella punkten och bilen i fråga. Värdet representerar bilens bidrag till tillgängligheten (bildensiteten) i den aktuella punkten och alla dessa värden summeras sedan för att tilldela värdet på tillgängligheten i punkten.

Värdet för B_x beror av avståndet enligt funktionen som illustreras i figur (6.1). Funktionen är baserad på tanken att en bil inom 100 m (d_c) kan anses vara så lättillgänglig den kan bli. För avstånd mellan 100 m och 500 m (d_w) avtar bilens tillgänglighet gradvis med

avståndet och när avståndet överstiger 500 m går tillgänglighetsbidraget asymptotiskt mot 0. Funktionen beskrivs i ekvation (6.16).

$$f(d) = \begin{cases} 1 & \text{om } d < d_c \\ \dfrac{1-p}{d_c^2 - d_w^2} d^2 + \dfrac{p d_c^2 - d_w^2}{d_c^2 - d_w^2} & \text{om } d \geq d_c \text{ och } d < d_w \\ \dfrac{p d_w^2}{d^2} & \text{i annat fall} \end{cases} \tag{6.16}$$

I (6.16) är d avståndet till bilen och p är en konstant som styr hur stor påverkan en bil har på långa avstånd. För stora värden på p avtar en bils tillgänglighetsbidrag snabbt med avståndet. I det implementerade systemet användes $p = 0.2$. Detta värde valdes godtyckligt; vilket värde på p som bäst beskriver hur tillgänlig en bil uppfattas på längre avstånd skulle behöva bestämmas genom simulering eller verkliga tester.



Figur 6.1. En bils bidrag till tillgängligheten i en punkt, som funktion av avståndet mellan bilen och punkten.

Den exakta beräkningsmetoden för tillgänglighet beskrivs i algoritm 1. Algoritmen har tidskomplexitet $O(nb)$, för n zoner i driftområdet och b lediga bilar.

I en praktisk implementation är det önskvärt att kunna ha förberäknade tabeller över önskade och faktiska värden på bildensitet. För att detta ska vara möjligt måste en zonindelning av det kontinuerliga geografiska området göras, så att varje zon kan representeras som en post i en tabell.

I det implementerade systemet har en zonindelning gjorts genom att dela in hela driftområdet i kvadrater med sidlängd 10 m, se figur 6.2a. Denna indelning får vissa oönskade konsekvenser. Exempelvis beräknas tillgänglighet och efterfrågan även för zoner som helt

befinner sig i vattendrag, vilket inte är optimalt. Mer sofistikerad zonindelning, som i figur 6.2b, skulle kunna göras för att undvika detta och andra liknande problem (t.ex. oländig terräng).

Algoritm 1 Interpolation av bildensitet över ett område. `LAT_ZONES` är antalet zoner området är indelat i längs latitudriktningen (och analogt för `LON_ZONES`). `ZONE_HEIGHT_DEGS` är storleken av en zon i grader latitud, och `ZONE_WIDTH_DEGS` är motsvarande i grader longitud. Funktionen f illustreras i figur 6.1

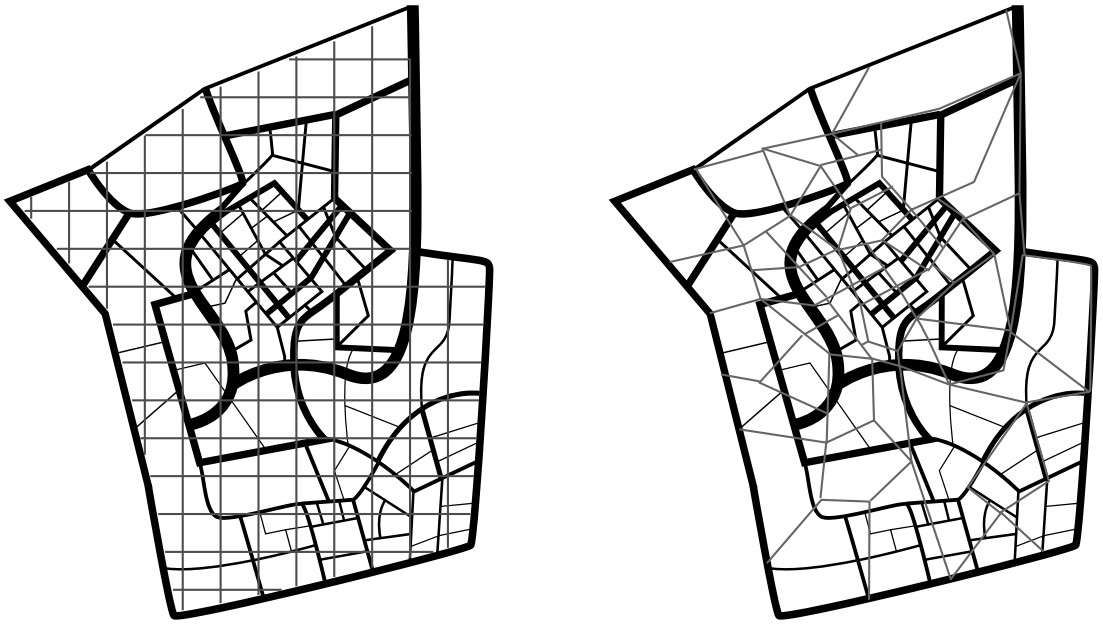
```
function densityInterpol(lats, lons, weights)
    density[LAT_ZONES][LON_ZONES]
    for ilat = 0 → LAT_ZONES do
        lat ← LAT_MIN + ilat * ZONE_HEIGHT_DEGS
        for ilon = 0 → LON_ZONES do
            lon ← LON_MIN + ilon * ZONE_WIDTH_DEGS
            val ← 0
            for i = 0 → lats.size do
                dval ← f(lat, lon, lats[i], lons[i])
                dval ← dval * weights[i]
                val ← val + dval
            end for
            density[ilat][ilon] ← val
        end for
    end for
    return density
end function
```

Demand prediction

Metoden för att beräkna efterfrågan liknar den som används för att beräkna bildensitet. Som kan utläsas ur algoritm 1 finns det inget som begränsar den till att endast kunna användas på bilar, och samma algoritm används också för att beräkna efterfrågan.

Beräkningen utgår från historiska bokningar. För alla historiska bokningar är positionen där bokningen gjordes känd. För att beräkna efterfrågan i en punkt beräknas ett avståndsberoende värde för alla bokningar enligt samma metod som används för beräkningen av bildensitet. Utöver det multipliceras varje boknings bidrag med en vikt som avtar med bokningens ålder. Denna viktning görs då nyare bokningar antas spegla aktuell efterfrågan bättre än äldre bokningar.

Användandet av historiska bokningar till att förutspå nuvarande bilbehov bygger på antagandet att behovet av bilar är periodiskt. Med ett antagande om veckolånga perioder betyder detta att bilbehovet på exempelvis onsdagar kommer att se likadant ut som bilbehovet gjorde under tidigare onsdagar. Att perioden av bilbehovet skulle vara just



(a) Enkel rutnätsindelning

(b) En mer komplex indelning där zonstorlekarna har anpassats efter biltäthet

Figur 6.2. Två varianter av zonindelning

en vecka är långt ifrån självklart. En så lång period ger också en sämre upplösning, då variationer kortare än en dag inte tas med i uppskattningen. En avvägning måste dock göras, då metoder som ger högre upplösning (exempelvis antagandet att alla onsdag förmiddagar är likadana med hänseende på bilefterfrågan) också ger färre bokningar att basera förutsägelserna på (en förmiddag är kortare än en hel dag), vilket gör förutsägelserna känsligare för slumpmässiga variationer.

6.2.2. Alternativ interpolation av behov baserat på hotspots

Utöver den modell för beräkning av bilbehov som beskrivs i avsnitt 6.2.1 utvecklades en modell där behovet inom driftområdet baseras på manuellt utplacerade *hotspots*. En hotspot definierades som en punkt med ett känt behov av bilar, exempelvis kan man tänka sig att det finns ett stort behov av bilar nära tåg- eller busstationer. Mellan dessa punkter interpoleras sedan bilbehovet baserat på det angivna behovet för varje hotspot. Metoden som användes för interpolation baserades på Shepards metod (Shepard, 1968).

Modellen baserad på hotspots förkastades då den kräver att hotspots placeras ut på ett lämpligt sätt, men ger inga metoder eller riktlinjer för hur detta ska göras. Till skillnad från modellen baserad på historiska bokningar kan den inte automatiskt anpassa sitt beteende när mer data om hur systemet används blir tillgänglig.

6.2.3. Alternativ metod för densitetsinterpolation

En alternativ metod för interpolation av bildensitet utvecklades. Metoden förkastades, men beskrivs här för fullständighetens skull.

Den alternativa interpolationsmetoden för bildensitet är mycket lik den metod som presenteras under avsnitt 6.2.1. Dock används en annan funktion för beräkning av en bils tillgänglighetsbidrag i en punkt, som funktion av avståndet mellan bilen och punkten. Metoden kan beskrivas med algoritm 1, men med en alternativ funktion f. Den alternativa funktionen är viktfunktionen som används i Shepards metod och är alltså på formen $f(d) = \frac{1}{d^p}$ (Shepard, 1968).

Denna alternativa interpolationsmetod konstaterades vara olämplig för beräkning av tillgänglighet. Tillgänglighetsvärden för områden som helt saknar närliggande bilar blir för höga eftersom varje bil får för stort globalt inflytande. Exempelvis kunde områden påvisas där tillgängligheten beräknades till 5–6 bilar trots att närmaste bil befann sig flera mil därifrån. Problem uppstår även när punkter ligger väldigt nära bilar, då $\frac{1}{d^p}$ blir mycket stort för små d. Dessa brister leder till att värdet på biltillgänglighet inte längre kan tolkas på något intuitivt sätt, till exempel "antal bilar inom gångavstånd", vilket är möjligt med den metod som beskrivs i avsnitt 6.2.1.

7. Resultat: Utvecklad mjukvaruprototyp

I detta avsnitt presenteras information om vilka mjukvarukomponenter som utvecklats, deras respektive användningsområden, funktionalitet och design. En översikt av designval följs av detaljerade beskrivningar av de ingående komponenternas struktur och uppgifter. Det bör noteras att ingen *strukturerad* enhets- eller användartestning genomförts, då mjukvaran utvecklades i undersöknings- och demonstrationssyfte.

7.1. Gränssnitt och protokoll

Klientapplikationerna (se avsnitten 7.5 och 7.6) är beroende av kommunikation med bokningssystemet. Denna kommunikation sker via väldefinierade gränssnitt i bokningssystemet, exponerade över HTTP.

Gränssnittet kan delas i tre delar baserat på åtkomstnivåer: administratörer, kunder och handhavare av optimeringssystem (*operatörsgränssnittet*). Syftet är att av säkerhetsskäl begränsa åtkomst till systemkritiska operationer för obehöriga användare.

7.1.1. Åtkomstnivåer

En användare med administratörsrättigheter kan genom gränssnitten, agera i en vanlig användares ställe. Detta är nödvändigt ifall en användare, i behov av att boka eller avboka en bil, på något sätt skulle förlora sin normala åtkomst till systemet. Administratörer kan även registrera nya bilar för användning och har åtkomst till all kunddata.

Kundgränssnittet tillgängliggör funktioner för sökning av bilar, bokning av tillgängliga bilar samt hämtning av egen användningsdata. Med undantag för bilsökningen kräver all interaktion med gränssnittet autentisering med användarnamn och lösenord kopplade till ett existerande aktivt konto i bokningssystemet.

7.1.2. Operatörs- och optimeringsgränssnitt

Operatörsgränssnittet används för att skicka data till och från de telematikenheter som finns monterade i bilpoolens bilar. Gränssnittet möjliggör regelbundna statusrapporter med information om bilens position, bränslenivå och laddningsstatus, men skulle troligtvis spela en begränsad roll i en verklig integration, då moderna telematiklösningar ofta inkluderar dylik funktionalitet (IRIS Technology, 2012; OnStar, 2012; Pilotfish, 2006).

För att tillgängligöra all data som potentiellt skulle kunna användas i optimeringsmoduler upprättades även ett gränssnitt för externa optimeringssystem, liknande det som beskrivs i avsnitt 7.4. Gränssnittet ger åtkomst till icke-konfidentiell data om historiska bokningar samt aktuella loggade fordonspositioner.

7.2. Modularitet

Systemet är modulärt i den mening att delsystem kan utvecklas separat och kopplas mot bokningssystemet utan att införa cykliska beroenden. Användningen av HTTP för utåtriktade gränssnitt innebär att specifika mjukvaruberoenden för anslutande klienter hålls till ett minimum. Detta gör det enkelt att bygga applikationer, exempelvis mobilklienter, som använder sig av systemets funktioner.

7.3. Bokningssystem

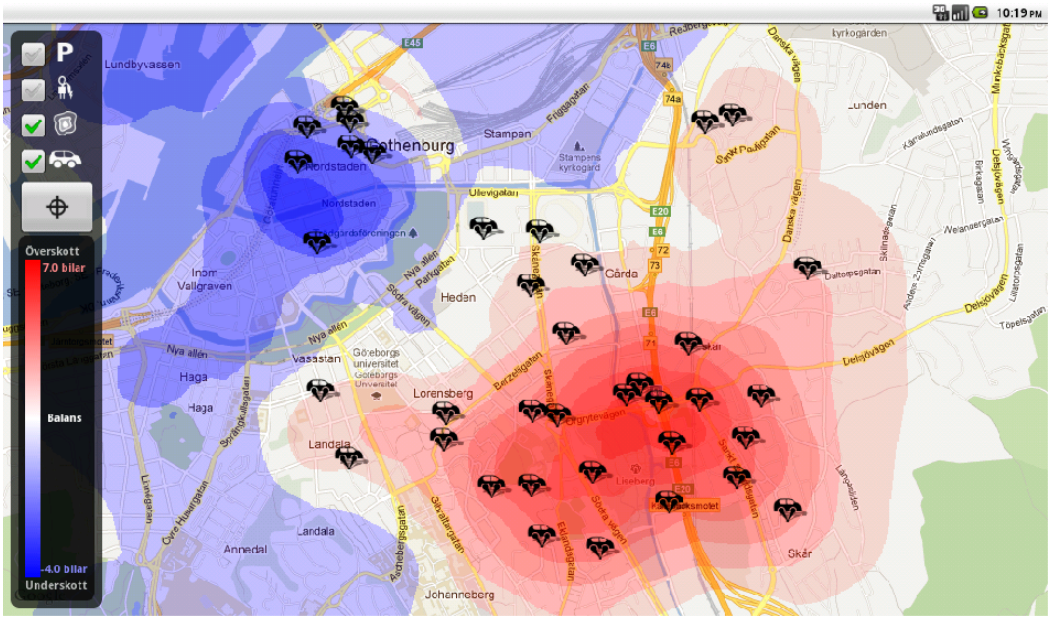
Bokningssystemet tillhandahåller funktionalitet för bokning av bilar samt information om bilpoolens tillstånd, exempelvis var lediga bilar finns. Bokningssystemet i sig har inget eget användargränssnitt, utan kommunicerar endast med andra program. Interaktion med användaren sker genom andra program (specifikt mobil- och surfplatteapplikationen). Bokningssystemets funktionalitet exponeras genom ett HTTP-baserat gränssnitt.

Bokningssystemet mottar kontinuerligt ny data om bilpoolens tillstånd. Bilpoolens bilar antas vara ständigt uppkopplade mot Internet för att skicka statusuppdateringar – position, batterinivå och huruvida batteriet för tillfället laddas – till bokningssystemet med regelbundna mellanrum. Det program som ansvarar för att skicka dessa uppdateringar är det program som körs på den bilmonterade surfplattan, beskriven i avsnitt 7.5.

För att ha möjlighet dra slutsatser kring och upptäcka mönster i bilpoolens användande, lagras all data som mottas av bokningssystemet permanent. Detta ger stora mängder data som kan undersökas statistiskt och användas för att identifiera resvanor och användningsmönster. Denna data skulle i framtiden i framtiden kunna användas till att utföra avancerad optimering av systemets drift.

7.3.1. Visualisering av fordonsbalans med värmekartor

Utifrån systemets aktuella balansvärden (se avsnitt 6.2) kan bokningssystemet rendera värmekartor, som kan hämtas via bokningssystemets webbgränssnitt. En värmekarta är en grafisk representation av data, där värden i en matris omvandlas till färgade pixlar. För graden av tillgång på bilar användes två distinkta nyanser, en för att indikera överskott och en för att indikera underskott. Graden av över- eller underskott (härefter benämnt som *avvikelse*) visualiseras genom att variera värmekartans opacitet.



Figur 7.1. Värmekarta som illustrerar över- och underskott placerad som ett lager ovanpå en karta. Placeringen av lediga bilar är också markerad. Bilden är en skärmbild från surfplatteapplikationen.

De värmekartor som bokningssystemet producerar är baserade på geografiska koordinater och kan placeras som ett lager över en (geografisk) karta. Ett exempel på hur detta kan se ut visas i figur 7.1.

Detta verktyg kan vara av nytta för att visualisera systemets tillstånd med avseende på fordonsfördelning och därmed kunna upptäcka problemområden med stort över- eller underskott. Om sådana områden upptäcks kan en operatör vidta åtgärder för att rätta till problemet, exempelvis manuella relokeringar. Ett exempel på en visualisering av ett problemområde finns i figur 7.1, där ett underskott av bilar kan ses nära figurens övre vänstra hörn.

Algoritmer

Värmekartan bygger på de värden för avvikelse från balanstillståndet som beräknas enligt de algoritmer som beskrivs i avsnitt 6.2.1. De värden som beräknas måste omvandlas till färger; det behövs alltså en funktion f : avvikelse $\rightarrow$ *ARGB*, där ARGB står för *Alpha-Red-Green-Blue*, vilket är ett system för att uttrycka färg (Red-Green-Blue) samt transparens (Alpha).

De beräknade värdena på avvikelse normaliserades i avsikt att förebygga överskridanden av yttre gränser i färgintervall och därmed undvika situationer där relativa skillnader blir svåra att utläsa. Två olika metoder för normalisering implementerades och utvärderades.

Den första metoden söker igenom alla beräknade avvikelsevärden och hittar det minsta (min) och det största (max). Sedan subtraheras min från varje värde, och det nya värdet multipliceras med $\frac{255}{max}$. På så sätt hamnar alla värden i intervallet $[0, 255]$. Dessa värden kan sedan konverteras till ARGB-färger genom uppslag i en tabell.

En andra normaliseringsalgoritm (*nollcentrerande*) togs fram för att underlätta avläsning av över- och underskott i relation till balans (avvikelse = 0), då den tidigare beskrivna algoritmen för normalisering inte har någon speciell färg som indikerar balanserade tillstånd.

Den nollcentrerande algoritmen normaliserar negativa och positiva värden separat. Utdata ligger i samma intervall $[0, 255]$, men ett värde på 127 (mitten av intervallet) indikerar alltid balans (avvikelse = 0). Denna metod gör det mycket tydligt vilka områden som har balans, över- eller underskott och är den normaliseringsmetod som använts i figur 7.1. Algoritmen för den nollcentrerande normaliseringen beskrivs i algoritm 2.

Algoritm 2 Den nollcentrerande normaliseringsalgoritmen. xs är det fält vars värden ska normaliseras till intervallet $[0, 2\,top]$.

```
function NORMALIZE(xs, top)
    pmax ← 0
    nmax ← 0
    for x : xs do
        pmax ← max(x, pmax)
        nmax ← min(x, nmax)
    end for
    nd ← 1
    pd ← 1
    if nmax < 0 then
        nd ← top/nmax
    end if
    if pmax > 0 then
        pd ← top/pmax
    end if
    for i = 0 → len(xs) do
        if xs[i] < 0 then
            xs[i] ← xs[i] * −nd + top
        else
            xs[i] ← xs[i] * pd + top
        end if
    end for
end function
```

Implementation och prestanda

Eftersom renderingen av värmekartor är för beräkningsintensiv för att utföras på mobila enheter görs det istället i bokningssystemet. Dock skulle även bokningssystemet snabbt bli överbelastat om värmekartan skulle renderas varje gång den efterfrågas. Istället renderas kartan endast en gång vid förändringar av bilpoolens tillstånd, till följd av bokningar eller avlämningar. Den lagras sedan som en PNG-bild och skickas till klienter vid förfrågan.

De värmekartor som bokningssystemet skickar över sitt HTTP-gränssnitt har korrekta cache-headers. Därför skickas inte bilder till klienter som redan har en aktuella värmekartor. Klienterna använder istället sin cachade version. Detta minskar trafikmängden från bokningssystemet och möjliggör pollande klienter.

7.3.2. Arkitektur

Bokningssystemet består av en webbserver som servar dynamiskt innehåll konstruerat från data hämtad ur en PostgreSQL-databas.

Under utvecklingens gång har databasen lagrats på samma dator som webbservern har körts på, men om så krävs skulle databasen kunna flyttas till en annan dator utan att någon ändring i programkoden krävs.

7.4. Optimeringsmodul

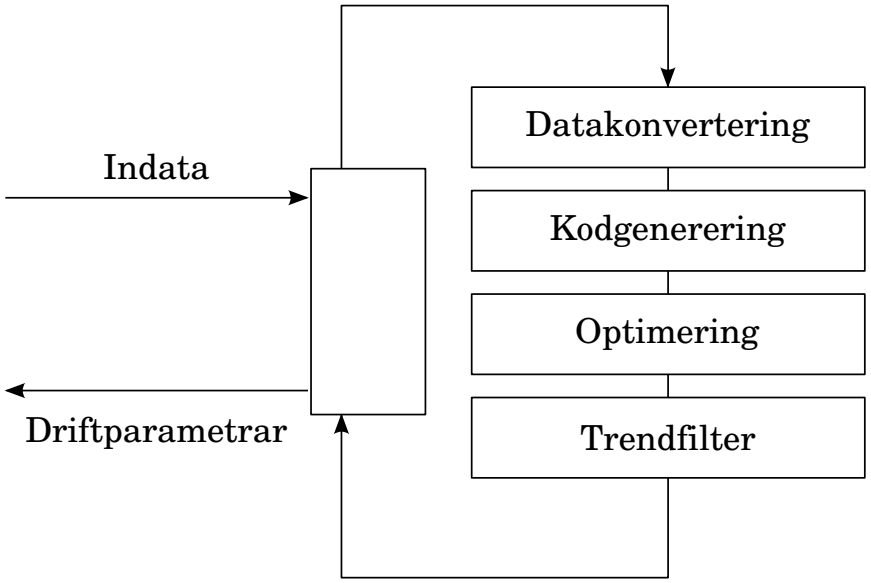
Optimeringsmodulen innehåller en fullständig implementation av Keks originalmodell för driftoptimering (se kapitel 3.3), vilken kan producera riktlinjer för framtagning av optimala arbetsscheman. Som ett visuellt hjälpmedel finns funktionalitet för att generera grafer som visar dessa optimala scheman (se figur 3.2 på s. 14 för en illustration av ett sådant schema). På grund av de tillkortakommanden som redogjordes för i avsnitt 4.3.2 implementerades inte den Kek-baserade modellen för dynamisk prissättning i den slutgiltiga prototypen.

7.4.1. Arkitektur

Kek-modellen utgör utgångspunkten för optimeringsmodulens struktur. På samma sätt som i Kek et al. (2009) användes följande sekvens av steg för optimeringsprocessen: *datakonvertering*, *kodgenerering*, *optimering* samt *trendfiltrering*. För varje steg skapades en komponent med en avgränsad uppsättning funktioner och en koppling mot nästföljande stegs komponent. Komponenterna listas nedan och deras inbördes kopplingar illustreras i figur 7.2.

- **Datakonverteringens** uppgift är att hämta obehandlad data och överföra den till ett internt format för fortsatt behandling. Överföringen sker genom mellanlagring i CSV (Comma Separated Values).

- **Kodgenereringen** formulerar ett optimeringsproblem i GMPL eller OPL, baserat på en definierad modell och indata från datakonverteringen, och skriver ut resultatet till en kodfil.
- **Optimeringen** består av ett anrop till en extern linjärprogrammeringslösare, `glpsol` eller CPLEX, med kodgenereringens utdata. Resultatet av optimeringen skrivs ut till en fil som nästföljande steg i optimeringsflödet kan läsa.
- **Trendfiltreringen** beräknar riktlinjer för bilpoolens drift baserat på utdata från optimeringssteget.



Figur 7.2. Översikt av de olika stegen i optimeringsmodulen.

7.5. Android-applikation för smarttelefoner

Mobilapplikationen utgör det primära användargränssnittet mot bokningssystemet och tillhandahåller funktionalitet som gör det möjligt för en registrerad användare att söka, boka och avboka bilar. Vidare kan en användare ta del av sammanställd information om sina tidigare bokningar, såsom sammanlagda kostnader och hur lång tid enskilda biltyper har använts.

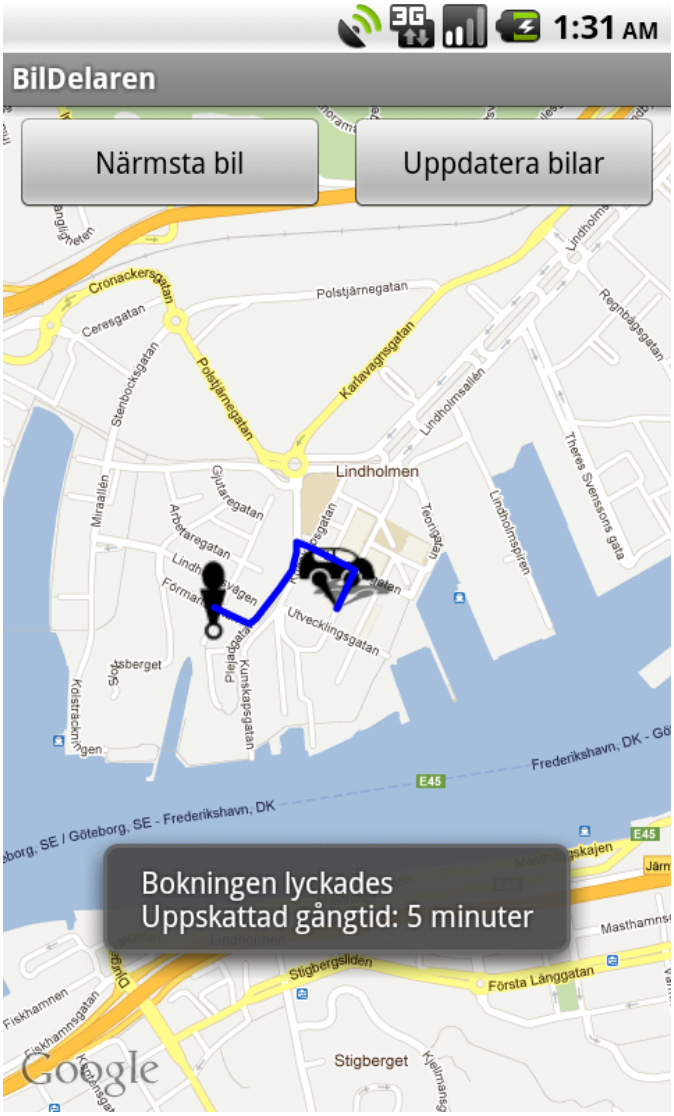
Så kallade *tillgångslarm* kan registreras för att varna användaren om antalet tillgängliga bilar inom en specifik radie understiger ett bestämt värde. Dessa larm hanteras av en systemregistrerad tjänst, vilket innebär att en varning kan visas även om applikationen inte körs. Larmen är användbara för kunder som inte har en bestämd tid för avfärd, men likväl måste få tillgång till en bil inom ett visst tidsintervall.

Applikationen är huvudsakligen en tillståndslös klient som arbetar mot bokningssystemet. En minimal mängd information om användarkonton lagras dock lokalt, för att undvika upprepad inmatning av användarnamn och lösenord vid de interaktioner mot bokningssystemet som kräver autentisering.

7.5.1. Gränssnittsdesign

Det grafiska gränssnittet utformades till ett praktiskt sök- och navigeringshjälpmedel, då användarens primära mål är att hitta och boka bilar. Därför är alla vyer byggda kring en central sökvy (se figur 7.3 på s. 48) som består av en navigerbar karta och tillhörande kontroller. Sökvyn har tre lägen: bilsökning, aktuell bokning och larmsättning. Möjligheten att byta till vissa lägen är begränsad: vyn för aktuell bokning, som inkluderar kontroller för att avboka, kan inte nås om en bokning inte pågår.

Sökvyns karta har en uppsättning lager med markörer som indikerar användarens nuvarande position i relation till olika intressepunkter. Bland dessa punkter ingår tillgängliga bilar och – om användaren har en aktiv bokning – den egna bokade bilen och en vägbeskrivning till den. Kartan används även för att skapa och konfigurera tillgångslarmen.



Figur 7.3. Mobilapplikation för Android. När användaren letat upp och bokat en bil så ges en vägbeskrivning tillsammans med uppskattad gångtid.

7.6. Android-applikation för bilmonterad surfplatta

Utvecklingen av applikationen utfördes på grund av nödvändigheten att kunna presentera prisincitament på ett tillgängligt och lättförståeligt sätt under resans gång, samt att låta kunder avrapportera huruvida bilar befinner sig i acceptabelt skick vad gäller städning och yttre skador vid upphämtning. I applikationens uppgifter ingår också att periodvis rapportera bilens status till bokningssystemet.

7.6.1. Gränssnittsdesign

Det grafiska gränssnittet är byggt med utgångspunkt från en central kartvy, liknande den som används i mobilapplikationen. Inga grafiska komponenter för autentisering finns, då föraren antas ha styrkt sin identitet genom bokningssystemet, innan han ges åtkomst till surfplattan.

Gränssnittet togs fram enligt idén att föraren måste kunna utläsa avlämningskostnader utan att minska sin uppmärksamhet på vägen. Som en följd har alla nödvändiga operationer gjorts tillgängliga genom stora och tydliga GUI-kontroller. Komplicerade processer, som bekräftelse av bilstatus vid början av varje resa, utförs i flerstegsprocesser med enkla delsteg.

På kartan finns fyra valbara lager för att presentera information. Urvalet av tillgänglig information är begränsat till vad som ansågs vara relevant för förarens val av destination och som även har ett praktiskt värde för körval under pågående resor. Varje lager aktiveras med en knapp, som kan ses i det övre vänstra hörnet av figur 7.1 på s. 43. Lagren presenteras i listan nedan.

- Parkeringsplatser visas som ikoner med färgnivåer som indikerar ungefärlig tillgänglighet. Data hämtas från Göteborg Stads trafikkontors webbservice och uppdateras periodvis för att ge föraren möjlighet att göra ett val av parkeringsplats utifrån både avstånd till destination och möjligheten att hitta en ledig parkeringsruta.
- Laddstationer visas som klickbara ikoner. Vid markering visas detaljerad information om laddplatsens adress/placering samt antalet laddningsuttag.
- Prisinformation i form av en semitransparent värmekarta, med en tillhörande prisskala för tolkning av kartan, kan aktiveras för att ge en uppfattning om hur valet av avlämningsplats inverkar på resans kostnad.
- Lediga bilar visas som ikoner på kartan. Syftet med detta är att en kund ska kunna byta bil om den nuvarande får slut på batteri.

7.6.2. Integration med bilens hårdvara

Kopplingen mellan surfplattan och resten av systemet begränsades till kommunikation över Internet. En verklig implementation skulle behöva omfatta en fysisk koppling mot en integrerad telematikenhet för att inte begränsa möjligheten att använda fordonet vid situationer då en Internetanslutning inte kan upprättas.

8. Diskussion

Detta kapitel presenterar en kritisk diskussion rörande påvisade begränsningar i resultatet av detta arbete. Det diskuteras hur dessa begränsningar inverkar på användbarheten av framtagna modeller och utvecklad mjukvara, samt hur de möjligen kan elimineras.

8.1. Mjukvarans användbarhet

Mjukvaran som utvecklades var aldrig tänkt att användas i verklig drift, och därför togs ingen hänsyn till frågor som säkerhet och juridiska begränsningar. Sådana frågor skulle ha störst inverkan på bokningssystemet, där användardata hanteras och priser beräknas. Vad gäller just prisberäkningen så kan det hända att metoden som används inte är laglig, då kunden inte nödvändigtvis kan bli informerad om faktiska kostnader för upphämtning och avlämning. En lösning skulle vara prisgarantier vid bokning och avlämning, men det skulle kräva betydande ändringar i bokningssystemet, och utökad funktionalitet i android-applikationerna.

Mobilapplikationen fungerade tillfredsställande för sitt syfte, men dess allmänna användbarhet kan diskuteras då den inte testats på mer än tre olika enheter med snarlik upplösning och prestanda. Grafiken är uteslutande skapad i vektorformat, och kan således enkelt renderas till varierande upplösningar för att anpassa applikationen till olika skärmstorlekar. Prestandan borde heller inte vara något problem med tanke på avsaknaden av grafiskt tunga operationer. Applikationens största brist är snarare användarupplevelsen, som inte fick något utrymme under utvecklingsprocessen.

Surfplatteapplikationen skulle primärt behöva utökas med ett gränssnitt för koppling mot telematikenheter, för att vara fullt användbar. Dessutom skulle uppritningen av värmekartan behöva optimeras och förbättras. Den bitmapsuppritning som används resulterar i att uppritade zoner, vid hög inzoomning, inte stämmer överens med sina reella geografiska motsvarigheter. Bitmappen som surfplatteapplikationen mottar från bokningssystemet är alltså för lågupplöst för den högsta möjliga inzoomingen. Denna brist kan tänkas inte enbart förvirra användare, utan också sänka kunders uppfattning om systemets tillförlitlighet.

8.2. Multipla driftområden

I detta arbete har undersökningarna begränsats till bilpooler med ett enda driftområde. En bilpool med fler än ett driftområde – ekvivalent med ett driftområde delat i disjunkta delar – innebär ytterligare svårigheter vad gäller driftoptimering, men kan också medföra nya möjligheter. Om kunder ska tillåtas förflytta bilar mellan två driftområden, kan det vara nödvändigt att införa begränsningar som sträcker sig utanför optimeringen. Det kan visa sig lönsamt att erbjuda särskilda prispaket som möjliggör pendling mellan driftområden. Det är möjligt att dessa erbjudanden då kan utformas så att dessa resor balanserar bilfördelningen mellan områdena.

8.3. Priselasticitet

Att uppskatta till vilken grad prisincitament (avgifter och rabatter) kan påverka en kunds beteende är centralt för teorin kring dynamisk prissättning. Givet ett visst prisincitament måste kundens villighet att offra bekvämlighet, exempelvis genom att gå en längre sträcka för att hämta en bil, kunna uppskattas. Snarare än den klassiska definitionen av priselasticitet, där efterfrågan artar sig i antalet konsumerade enheter, måste begreppet här omfatta hur långa sträckor en kund är villig att förflytta sig för att hämta en bil, samt hur långt ifrån sin slutdestination en kund är villig att lämna sin bil. I det här arbetet har priselasticitet förenklats till en funktion av efterfrågan på kostnad, utan hänsyn till avståndet. Funktionen används enbart i den utvidgade Kek-modellen, presenterad i avsnitt 6.1.

9. Slutsatser

Det visade sig kräva omfattande ändringar för att utöka Kek-modellen till att inkludera dynamiska priser. Detta gjorde det faktum att Kek-modellen verifierats fungera, i det sammanhang den ursprungligen konstruerats för, mindre intressant. Vi konstaterar att en tidsstegsbaserad linjärprogrammeringsmodell som använder variabler för att representera en högupplöst zonindelning kommer att ge orimliga beräkningstider för stora driftområden. Inget utlåtande görs om huruvida linjärprogrammering i allmänhet är lämpat för att modellera friflytande bilpooler.

Jämfört med erfarenheterna från Kek-modellen var implementeringen av dynamisk prissättning utifrån balansmodellen förhållandevis enkel. Givet att modellerna för tillgång och efterfrågan stämmer tillräckligt bra överens med verkligheten anser vi att balansmodellen skulle ge en rimlig representation av systemets balans. Även om prissättningen inte har avsedd effekt så skulle värmekartor som visualiserar balansvärden istället kunna användas för att ge operatörer en överblick av systemets tillstånd.

Vad gäller mjukvaruprototypen, utvecklades ett IT-system som kan användas med de utvecklade modellerna för dynamisk prissättning. I prototypen ingick, utöver implementationen av modellerna, ett primitivt bokningssystem och två kundgränssnitt i form av en mobilapplikation och en surfplatteapplikation. Eftersom bokningssystemet och applikationerna fungerade utan problem, när de demonstrerades vid Viktoriainstitutet, kan de sägas ha uppfyllt sina syften. Dock kan inget definitivt utlåtande om systemens kvalitet och användbarhet göras, då inga organiserade enhetstester eller användartester genomfördes.

9.1. Riktlinjer för framtida arbeten

Friflytande bilpooler är ett nytt koncept; car2go, grundat 2009, är det första kommersiella exemplet på en friflytande bilpool (Firnkorn och Müller, 2011). Därför är området "driftoptimering i friflytade bilpooler" relativt outforskat, till skillnad från den stora mängd forskning som gjorts om driftoptimering i stationsbaserade bilpooler.

Mer forskning behövs på området i allmänhet, men i detta kapitel presenteras några problem som upptäcktes under detta arbetes gång som bedöms speciellt intressanta att utforska ytterligare.

9.1.1. Användning av Kek-modellen

Ett antal förenklingar och begränsningar i Keks modell identifierades, utöver de som redogjorts för i Kek (2006). En av begränsningarna i bivillkoren innebär att två eller fler anställda i bilpoolen inte kan dela på en bil, någonting som skulle kunna vara en användbar strategi vid förflyttningar. En förenkling som identifierades är att antalet framtida avlämningar inte påverkas av att kunder nekas plocka upp bilar.

Inget försök gjordes i detta arbete att åtgärda dessa förenklingar i Keks modell, eftersom de riktlinjer modellen ger verifierats vara gynnsamma för Honda Diracc (Kek et al., 2009, s. 128). Det kan dock vara värt att begrunda i framtida arbeten.

Integration av dynamisk prissättning

Som beskrivet i avsnitt 6.1, verkar det inte möjligt att integrera dynamisk prissättning i modellen. Att integrera dynamisk prissättning i Keks modell skulle kunna fungera för stationsbaserade system, förutsatt att de problem som presenterats kan åtgärdas eller undvikas. En möjlig lösning skulle vara att använda en MIQP-lösare som kan hantera icke-konvexa målfunktioner.

För att kunna använda Kek-modellen som grund för dynamisk prissättning i ett stationsbaserat system skulle dock inverkan av stationers geografiska positioner behöva beskrivas. Eventuellt skulle modellen kunna utvidgas för att ta hänsyn till detta.

Probabilistiskt beteende

Om det skulle uppstå problem med att utgå från deterministiskt beteende i bilpooler, vilket Kek-modellen gör, skulle exempelvis förutspådd efterfrågan istället kunna uppskattas med sannolikhetsfördelningar för varje kombination av tid och station i modellen.

Riktlinjerna som ges av Keks modell skulle kunna vara mindre användbara än dylika riktlinjer från probabilistisk optimering. En modifierad version av Keks modell, som använder sannolikhetsfördelningar, har tidigare tagits fram (Nair, 2010). Denna modell har inte undersökts närmare i detta arbete, men den probabilistiska aspekten bör förmodligen beaktas i framtida arbeten.

9.1.2. Användning av fordonsbalans-baserad dynamisk prissättning

Det är oklart om fordons-balanserad dynamisk prissättning är effektiv i praktiken. Då denna modell är oprövad, vore det värdefullt att sätta den på prov genom simulering och verklig testning.

Nuvarande förenklingar och möjlig vidareutveckling

Metoden för beräkning av bildensitet skulle kunna vidareutvecklas för att också ta hänsyn till förutspådda värden för balansavvikelse. Till skillnad från den Kek-baserade metoden, ser denna endast till den aktuella situationen, vilket kan ha icke önskvärda konsekvenser.

Ett exempel på en icke önskvärd konsekvens är en situation där många bilar finns samlade vid en arbetsplats, strax innan många slutar sina jobb för dagen och vill åka hem. Ett stort bilbehov kommer att uppstå vid arbetsplatsen, men tiden före arbetsdagens slut är behovet relativt litet och balansmodellen skulle detta visas som ett överskott. Systemet skulle rekommendera att bilar flyttas bort från arbetsplatsen, vilket skulle vara ofördelaktigt då bilarna inom en kort period skulle bli mycket eftertraktade (och därmed väl utnyttjade).

Problemet skulle kunna lösas om man bland de historiska bokningar som används i algoritm 1 inkluderar bokningar som inträffat efter det aktuella klockslaget (exempelvis bokningar som inträffat kring kl. 17:10). En manuell indelning av dygnet i lämpliga tidsintervall skulle kunna göras för att ta hänsyn till situationer liknande den som beskrivs ovan.

Modellen tar ej heller direkt hänsyn till bilpoolens lönsamhet. Endast balans beaktas, och det görs inga bedömningar om huruvida det alltid är lönsamt att uppnå balans. Experiment som följer upp detta arbete skulle kunna verifiera effektiviteten hos de modeller för kostnadsminimering och dynamisk prissättning som implementerats. Den teoretiska delen av detta arbete berör främst möjligheter och begränsningar hos utvecklade modeller för dynamisk prissättning, samt uppställning av relevanta riktlinjer för efterföljande arbeten. Ett lämpligt sätt att testa modellernas effektivitet vore möjligen trafiksimulering.[1]

9.1.3. Dynamisk prissättning i elbilspooler – ett ekonomiskt hållbart koncept?

Projektet har visat att konstruktionen av ett IT-system för dynamisk prissättning i friflytande bilpooler är praktiskt genomförbart. Eftersom fokus har legat på den tekniska sidan av problemet har inget försök gjorts att påvisa praktisk lönsamhet av dynamisk prissättning av det slag som beskrivs i denna rapport. En sådan utvärdering skulle kräva ett tvärvetenskapligt angreppssätt för att kunna undersöka användarbeteende i relation till långsiktig ekonomisk hållbarhet.

För att utföra en realistisk utvärdering måste faktorer inräknas som relaterar till kostnaden att konstruera och underhålla IT-system som krävs för att använda dynamisk prissättning i praktiken. Sådana ekonomiska beräkningar skulle kunna påvisa att positiva effekter av dynamisk prissättning inte väger upp initiala investeringskostnader. Vi hoppas dock att dynamisk prissättning visar sig vara lönsamt, och att denna rapport kan vara ett bidrag till dess utveckling.

[1] Användning av simuleringsverktyget SUMO (`http://sumo.sourceforge.net`) övervägdes i början av projektet.

Litteraturförteckning

Distributed Systems Group (2012). *Dynamic One-way Car Sharing*. Trinity College Dublin. `http://www.dsg.cs.tcd.ie/FutureCities/SmartTransportation/DynamicOneWayCarSharing` (2012-04-20).

Firnkorn, Jörg och Müller, Martin (2011). *What will be the environmental effects of new free-floating car-sharing systems? The case of car2go in Ulm*. Ecological Economics, 70(8):ss. 1519–1528. ISSN 0921-8009. doi:10.1016/j.ecolecon.2011.03.014. `http://www.sciencedirect.com/science/article/pii/S0921800911001030` (2012-05-30).

Garthwaite, Josie (2011). *Car2go, Daimler-Backed Sharing Program, to Go Electric in San Diego*. Wheels, The New York Times. `http://wheels.blogs.nytimes.com/2011/07/13/car2go-daimler-backed-sharing-program-to-go-electric-in-san-diego/?emc=eta1` (2012-06-01).

Granovskii, Mikhail, Dincer, Ibrahim, och Rosen, Marc A (2006). *Economic and environmental comparison of conventional, hybrid, electric and hydrogen fuel cell vehicles*. Journal of Power Sources, 159(2):ss. 1186–1193. ISSN 0378-7753. doi:10.1016/j.jpowsour.2005.11.086.

Helmersson, Dicte (2012). *Bilpool*. Nationalencyklopedin. `http://ne.se/lang/bilpool` (2012-02-26).

Internet Engineering Task Force (1999). *RFC 2617: HTTP Authentication: Basic and Digest Access Authentication*. Teknisk rapport, The Internet Society. `http://tools.ietf.org/html/rfc2617` (2012-04-26).

IRIS Technology (2012). *IRIS 2002 product sheet*. IRIS Technology. `http://iris-technology.co.uk/pdfs/IRIS1002.pdf` (2012-04-27).

Jensen, Justin (2010). *A Balancing Act: the Future of Car Sharing and Driving-as-a-Service*. MIT Entrepeneurship Review. `http://miter.mit.edu/node/168` (2012-04-27).

Karlberg, Lars Anders (2011). *Sladdlös laddning bäst för elbilarna*. Ny Teknik. 8 September.

Kek, Alvina G.H (2006). *Decision support tools for carsharing systems with flexible return time and stations*. Doktorsavhandling, National University of Singapore.

Kek, Alvina G.H., Cheu, Ruey Long, Meng, Qiang, och Fung, Chau Ha (2009). *A decision support system for vehicle relocation operations in carsharing systems*. Transportation Research Part E: Logistics and Transportation Review, 45(1):ss. 149–158. ISSN 1366-5545. doi:10.1016/j.tre.2008.02.008.

Makhorin, Andrew (2010). *GNU Linear Programming Kit, Reference Manual for GLPK Version 4.45*. Free Software Foundation.

MIT Smart Cities Group (2010). *Mobility-on-Demand*. Smart Cities. `http://cities.media.mit.edu/projects/mobilityondemand.html` (2012-04-26).

Mitchell, William J, Borroni-Bird, Chris E, och Burns, Lawrence D (2010). *Reinventing the Automobile: Personal Urban Mobility for the 21st Century*. The MIT Press. ISBN 978-0-262-01382-6.

Nair, Rahul (2010). *Design and analysis of vehicle sharing programs: A systems approach*. Doktorsavhandling, University of Maryland.

OnStar (2012). *OnStar*. `http://onstar.com` (2012-04-27).

Oracle Corporation (2012). *Jersey*. java.net. `http://jersey.java.net` (2012-04-24).

Pilotfish (2006). *Telematik för nya affärsmöjligheter*. White paper. `http://pilotfish.se/index.php?id=downloads`, välj "White Paper on Telematics (In Swedish)" (2012-05-29).

Shepard, Donald (1968). *A two-dimensional interpolation function for irregularly-spaced data*. I *Proceedings of the 1968 23rd ACM national conference*, ACM '68, ss. 517–524. ACM, New York, NY, USA. doi:10.1145/800186.810616.

Skyhook (2012). *SpotRank Overview*. Skyhook. `http://skyhookwireless.com/location-intelligence` (2012-05-13).

Sommerville, Ian (2010). *Software Engineering, Ninth Edition*. Pearson. ISBN 978-0-13-705346-9.

Stenberg, Daniel (2012). *libcurl bindings*. cURL. `http://curl.haxx.se/libcurl/bindings.html` (2012-04-27).

Tan, Christopher (2008). *End of the road for Honda car-sharing scheme*. The Straits Times. Feb 29.

The Hertz Corporation (2012). *Vanliga frågor och svar*. Hertz Freerider. `http://hertzfreerider.se/unauth/display_faq.aspx` (2012-05-13).

Transport Canada (2012). *Bike sharing guide*. Government of Canada. `http://tc.gc.ca/eng/programs/environment-urban-guidelines-practitioners-bikesharingguide2009-3-1662.htm` (2012-04-26).

U.S. Department of Energy (2011). *Comparing Energy Costs per Mile for Electric and Gasonline-Fueled Vehicles*. Teknisk rapport, U.S. Department of Energy.

Viktoriainstitutet (2012). *About Us*. `http://viktoria.se/aboutus_vi` (2012-04-20).

Williander, Mats (2011). *Nya miljöbilspremien – bortkastade pengar*. Göteborgsposten. `http://gp.se/nyheter/debatt/1.708204-nya-miljobilspremien-bortkastade-pengar` (2012-04-27).

A. Komplexitetsanalys av algoritm för Kek-baserad dynamisk prissättning

Den tidskomplexitetsanalys som presenterades i avsnitt 5.3.1 finns bifogad i sin helhet med start på följande sida. Arbetet undersöker effektiviteten av den utökade Kek-modellen och redogör för mätningar som indikerar att algoritmen ej kan anses användbar.

Computational time complexity analysis of a one-way carsharing MILP model

Roland Hellström Keyte
TDA517
Chalmers University of Technology

May 22, 2012

The asymptotic computational time complexity of a carsharing MILP model used with the optimizer `glpsol` was determined in the number of stations involved in the model. The time complexity proved to be exponential in computers with limited RAM available. Consequently, the MILP model was discarded for use in a bachelor's project in which there was a need to solve the model using a large number of stations.

Introduction

In one-way carsharing services, the carsharing customers are granted the additional freedom of not having to return the rented cars to the pick-up location. This extra freedom generally leads to pile-ups of cars in certain areas, thus causing a deficit of cars in other areas (Barth 1999).

To satisfy customer demands for cars in all areas, carsharing staff are normally required to relocate cars from areas with a surplus of cars to areas with a deficit. These relocation operations are generally time consuming and expensive. An optimizer can be used to find cost efficient strategies for carrying out these relocations (Kek 2006).

The analysis presented in this report was conducted for a bachelor's project in which there was a need for finding such relocation strategies, but for one-way carsharing services using a very large number of stations. Doing this is problematic since an optimizer takes more time finding an optimal relocation strategy the more stations there are (which is later demonstrated in Figure 1).

The computational time complexity analysis was needed in order to get an idea of how many stations a MILP (Mixed Integer Linear Programming) optimizer could handle within a reasonable amount of time. The time complexity was sought instead of the actual time since the time much depends on the hardware used. The time complexity, on the other hand, is a property of the mathematical model used by the optimizer to describe the relocation problem (a MILP model in this case), and the optimization algorithm (Papadimitriou 1998). A time complexity of $O(n^3)$ (the number of stations is denoted by n in this report) or less was hoped for, as this would have been sufficient for the MILP model to be useful in the bachelor's project.

Method

The MILP model, which was used in the bachelor's project to model the relocation problem mathematically, was initially coded into the mathematical programming language GMPL. A time complexity analysis tool was subsequently coded in Java. The tool measured the solving times taken for the chosen optimizer, `glpsol`, to find optimal relocation strategies for different number of stations (different values of n) in the GMPL program.

When the number of measurements was sufficient, the tool generated graphs showing the correlation between n and the computational solving time. The graphs were later visually analyzed in order to find appropriate functions forms to use for curve fitting. The function forms were developed iteratively through trial and error, until forms were found which could be accurately fitted to the measured solving times. The resulting function forms were finally used to determine the time complexity of the optimizer in n.

Measurements

The measurements made by the complexity analysis tool were taken using values of n ranging from 1 to 170. The data was collected separately for the total execution time and the execution time of the underlying branch-and-bound optimization algorithm[1]. The separation was made to enable a better understanding of the relationship between the optimizer and the time complexity. The measurements are shown in Figure 1, which was generated by the analysis tool. As depicted in the figure, the optimization time taken clearly increases with n.

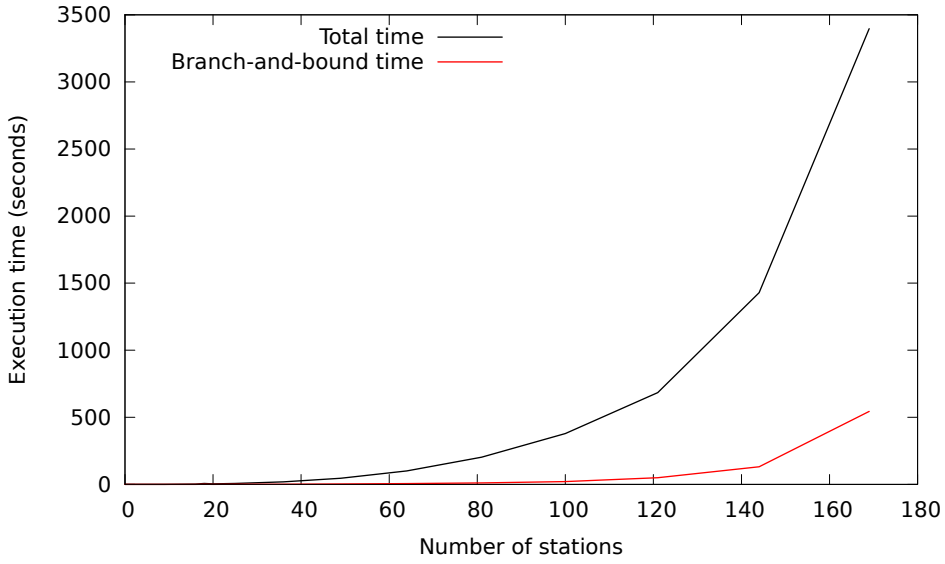


Figure 1: Solving time of `glpsol` with the MILP model, as a function of n

To avoid time-consuming calculations the analysis tool was designed to exponentially increase the size of the steps between measurements, which explains why the curves have lower resolution for higher values of n in Figure 1.

[1] `glpsol` uses the branch-and-bound and the simplex algorithms to optimize these types of models (Makhorin 2010)

All measurements were averages of 2-3 optimization runs. Generally, measurements taken using the same value for n were about the same, which meant averages from large populations were not needed. The computer which was used had a 3.6 GHz Intel i5 2500K CPU and 8GB of DDR3 PC12800 SDRAM. To produce accurate measurements, the computer was left to work exclusively on the time complexity analysis during all optimization runs. Moreover, the measurements were taken sequentially to guarantee the independency between them.

Curve fitting

By fitting functions to the plotted data in Figure 1, these fits could be used to determine the computational time complexity. The complexity class was determined by looking at the asymptotically dominant terms of the fitted functions. For instance, the time complexity would have been said to be $O(n^3)$ (third degree polynomial) if the dominant term was an^3 (with a being a constant) asymptotically in the fitted function. The plotting tool `gnuplot` was used for the curve fitting.

The forms of the fitting functions were found through trial and error, and when the fitted functions visually matched the measured data in Figure 1, the forms were considered acceptable. The constant term was omitted in all tested function forms since the time taken to find an optimal strategy can be assumed to be instantaneous for $n = 0$.

The shape of Figure 1 led to the initial guess that the time complexity was either $O(n^3)$ or $O(a^n)$.[2] Figure 2 shows fits with a third degree polynomial function form.

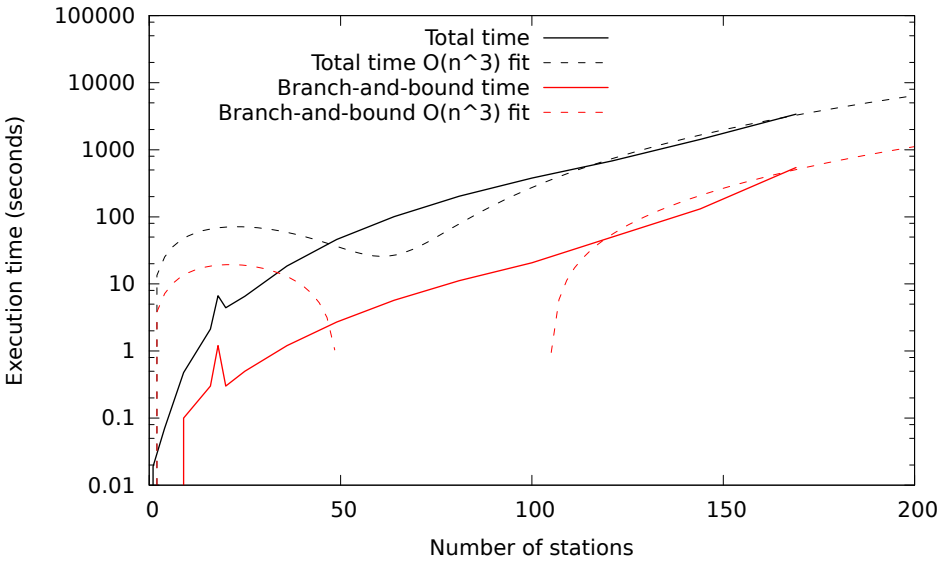


Figure 2: Solving time as a function of n and fitted $O(n^3)$ curves

A logarithmic scale is used in Figure 2, which makes the inaccuracies of the fits very apparent. Additionally, since the curves of the measured data are not straight in this scale, it is clear that the time complexity in n is not exponential. The time complexity could therefore be assumed to be exponential with a polynomial term dominant for low values. However, on the computer used, the RAM (Random Access Memory) required for the optimization increased beyond the available RAM for approximately $n > 100$. This led to the hypothesis that the

[2]These are also common complexity classes for the branch-and-bound algorithm (Thakoor 2010)

time complexity is in fact polynomial when only utilizing the RAM, but exponential when also employing the hard drive[3].

To test this hypothesis, polynomial functions of different shapes were first fitted to the data points below 100. Then, exponential functions were fitted to the data points above 100. This eventually led to very good fits, where an $O(n^3)$ form proved very accurate for data points below 100, and the form ae^{bn} (a and b being constants) matched the measured data points above 100 very well (see Figure 3). Since the functions visually matched the data in Figure 1, the resulting $O(n^3)$ and $O(a^n)$ complexities were considered acceptable.

Result

The asymptotic computational time complexity turned out to be $O(n^3)$ in the case where an unlimited amount of RAM is available. With limited RAM, the time complexity was found to be $O(a^n)$. This can be seen in Figure 3, where the hard drive is used for approximately $n > 100$.

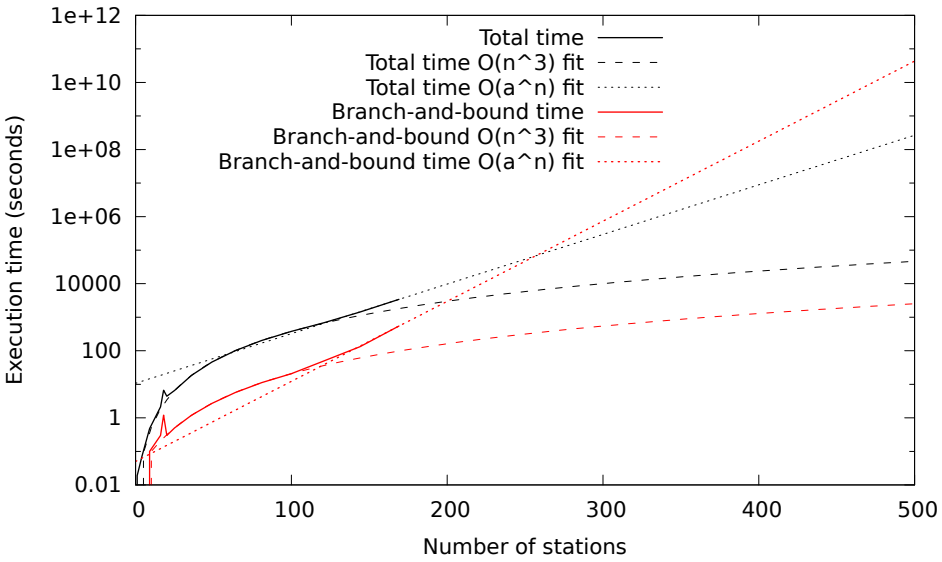


Figure 3: Solving time as a function of n and fitted curves

Conclusion

From the result of this analysis, the computational time complexity appeared to be exponential in n because of the limited memory available in the computer used. The result was something of a disappointment, since a time complexity of $O(n^3)$ was originally hoped for. Since the time complexity was $O(n^3)$ with unlimited RAM, using the MILP model with many stations might however still be an alternative for some applications. In the bachelor's project however, the model was discarded since more than 1000 stations were needed in the model. As seen from the extrapolation in Figure 3, solving the model for 500 stations would likely take between 10^8 and 10^{10} seconds without having a computer with unlimited RAM. This solving time translates

[3]In the operating system which was used (Debian GNU/Linux 3.2.0-2-amd64), the hard drive is used as virtual memory when no RAM is available

to approximately 30 years, which can be compared to the somewhat three hours it would take to solve the model with an unlimited amount of memory. Even though the carsharing MILP model was discarded, the result from this analysis proved valuable for the bachelor's project, as it triggered the development of a more computationally time-efficient model.

References

Barth, M., Todd, M. (1999) Simulation Model Performance Analysis of a Multiple Station Shared Vehicle System. *Transportation Research, Part C: Emerging Technologies.* Vol 7, pp. 237-259.

Kek, A. (2006) *Decision support tools for carsharing systems with flexible return time and stations.* Singapore: National University of Singapore. (A thesis submitted for the degree of doctor of philosophy. Department of civil engineering).

Makhorin, A. (2010) *GNU Linear Programming Kit Reference Manual for GLPK Version 4.45.* Draft. Boston: Free Software Foundation, Inc.

Papadimitriou, C. H., Steiglitz K. (1998) *Combinatorial Optimization: Algorithms and Complexity.* Unabridged edition. Dover Publications.

Thakoor N., Gao J. (2010) Branch-and-Bound for Model Selection and its Computational Complexity. *IEEE Transactions on Knowledge and Data Engineering.* To be published.