En simulerings- och optimeringsstudie på DSB S-tog i Köpenhamn.

Datum: 2009-06-12

Författare: Per Lindström, Per Nielsen

Handledare: Patrik Tydesjö, Stefan Vidgren

Förord

Detta examensarbete har skrivits i ett samarbete mellan DSB:s analysgrupp för S-tåg, och avdelningen för produktionsekonomi vid Lunds Tekniska Högskola. Kollektivtrafiken är idag en väldigt viktig del av en stads infrastruktur, och väldigt många människor är beroende av att den fungerar väl. Därför är det väldigt intressant att få skriva ett examensarbete på ett aktuellt ämne som berör många.

Vi hoppas att vår insats ska kunna medföra att fler tåg i Köpenhamn kommer och går i tid, och således att tåg blir ett attraktivare transportmedel för Köpenhamnsborna.

Ett stort tack till våra handledare Stefan Vidgren, och Patrik Tydesjö, som hjälpt oss med mycket. Tack också till Rudolf Abelin och Lars Telg på datordriftgruppen som hjälpt oss med tillhandahållande av beräkningsservrar. Slutligen vill vi också tacka våra kompisar och kursare på Lunds Tekniska Högskola som har varit till ovärderlig hjälp.

Sammanfattning

Titel: En simulerings- och optimeringsstudie av tågtrafiknätet på DSB S-tog AS i Köpenhamn

Författare: Per Lindström och Per Nielsen

Handledare: Patrik Tydesjö och Stefan Vidgren

Syfte: Förbättra regulariteten på Köpenhamns pendeltåg, S-tåg, genom simulering. Detta uppnås genom att simulera olika tidtabeller i en modell av S-tågnätet samt att utveckla en optimeringsheuristik som förbättrar möjligheten att simulera olika tågtidtabeller.

Avgränsning: Om än önskvärt så vore det praktiskt omöjligt att testa alla kombinationer av tider för tidtabellen. Därför har vi valt att endast testa de tider vi på goda grunder anser har möjlighet att bli optimala. Dagens tidtabell har således använts som grund, eftersom det ansetts orimligt att de värdena skulle vara helt fel. Simuleringsmodellen tar inte hänsyn till extraordinära fel som till exempel lok som går sönder, olyckor, väderfenomen, och så vidare.

Metod: Då vi utfört jämförande analys av antalet regularitetsbrott vid olika val av en isolerad faktor, är den valda metoden för att lösa problemet en experimentstudie, men med vissa modifikationer. Huvudsyftet är snarare av problemlösande natur än förklarande.

Slutsatser: Även om endast ett fåtal tidtabeller simulerats, har resultatet varit bättre än dagens tidtabell. Så länge modellen faktiskt beskriver verkligheten tillräckligt bra, borde resultatet kunna användas för applicering på S-tågnätet.

Nyckelord: Simulering, optimering, statistik, hierarkisk planering

Abstract

Title: A Simulation and Optimization study of the train network at DSB S-tog AS in Copenhagen.

Authors: Per Lindström and Per Nielsen

Supervisors: Patrik Tydesjö and Stefan Vidgren

Purpose: To improve the regularity of the Copenhagen commute train, S-tog, by simulation. This is achieved by simulating different time schedules in a model of the S-tog network, and to develop an optimization heuristic that improves the ability to simulate different time schedules.

Delimitation: Even though desirable, it would be practically impossible to try out every single time schedule combination. Consequently we have chosen to only test those time schedules that we believe have a good possibility of being optimal. The time schedule of today has therefore been used as a base, since it would be unreasonable that these values are completely wrong. The simulation model does not take in consideration extraordinary events, such as weather phenomena, and train malfunctions.

Method: Since we have conducted a comparative analyse of the regularity for different settings of an isolated factor is the chosen method called an experimental study, but with some modifications. The main purpose is rather of a problem solving nature, than explaining.

Conclusions: Even though we only simulated a small part of the total amount of different time schedules, we achieved a result that was better than the time schedule used today. As long as the developed model describes the reality good enough, the result should be useful for application on the S-tog train network.

Key words: Simulation, Optimization, Statistics, Hierarchical Planning.

Innehåll

1 Inledning

Inledningen är till för att orientera läsaren i vad arbetet innehåller. I detta inledande kapitel ges en kort bakgrund av uppdragsgivaren samt uppgiften. Problematisering, syfte och avgränsningar till uppgiften presenteras. Läsaren rekommenderas att iaktta underkapitlet om definitioner.

1.1 Bakgrund[1]

DSB, Danske Statsbaner, är ett företag ägt av danska staten som står för tågtrafiken i hela Danmark. Trots att det ägs av staten fungerar det som ett vinstdrivande företag, med de krav som det innebär. DSB har cirka 9000 anställda, och omsätter årligen omkring 10,5 miljarder danska kronor. S-tågen, som är en del av DSB:s verksamhetsområde, fungerar som pendlingståg i Storköpenhamn. S-tåg nätet används dagligen av cirka en kvarts miljon köpenhamnare och ditresta människor. Således ligger det i mångas intresse att tågen kommer i tid.

Vi blev introducerade för DSB via Stefan Vidgren, före detta anställd på Lunds Tekniska Högskola, numera anställd hos DSB som operationsanalytiker på deras analysgrupp för S-tåg. Han hade tidigare handlett ett examensarbete för DSB där studenter hade tagit fram en modell för S-tågnätet. Nu önskade han två examensarbetare med goda kunskaper inom simulerings - och optimeringsområdet som kunde ta vid där de tidigare slutade. Arbetets natur och den dagsaktuella uppgiften gjorde att vi kände oss manade att ta oss an detta problem.

1.2 Problematisering

För en stad av Köpenhamns storlek är en fungerande allmän kommunikation en viktig del av den totala infrastrukturen. Därför har DSB utformat fyra fokusområden[2]:

[1] http://www.dsb.dk/Om-DSB/Virksomheden/ 2009-04-22
[2] Projektplan för BuffertSIMS fas 2 av Stefan Vidgren 2009-01-22

- Fler kunder
- Tåg i tid
- Bättre image
- Effektivare verksamhet

Analysgruppen har därefter definierat arbetsområden för tågnätverket som stödjer dessa fokusområden:

- Bättre tjänster
- God återhämtningsförmåga
- Robusthet

Det viktigaste av de ovanstående är att tågen kommer i tid, då det snabbt återspeglar sig i de övriga områdena. Fler tåg i tid ger bättre image och fler kunder. Ett sätt att uppnå detta är att utforma en bra tidtabell som ger minsta möjliga försening. Tyvärr är det svårt att testa tidtabeller i verkligheten, och därför vill man använda sig av simulering som är ett mycket kraftfullt verktyg. Genom att göra en modell av verkligheten och testköra olika värden kan man förhoppningsvis hitta en optimal tidtabell. En modell skapades redan 2008, av två examensarbetare från Lunds Tekniska Högskola, Björn Johnsson och Patrik Bjärkeson. Dock är det inte möjligt att testa alla möjliga tidtabeller, eftersom det blir enormt många.

Vår uppgift är därför att finna en simuleringsheuristik, som har till uppgift att minska antalet teoretiska tidtabeller, och sedan simulera dessa, för att förhoppningsvis hitta en tidtabell, som ger mindre försening än dagens tidtabell. Modellen tar ganska lång tid att köra. Det var därför önskvärt att dra ner på den tiden för att möjliggöra test av fler tidtabeller.

1.3 Syfte och målsättning

Utveckla en simuleringsheuristik, som reducerar antalet teoretiska tidtabeller till ett rimligt antal, och därefter simulera dessa för att i slutändan förhoppningsvis finna en bättre tidtabell än idag. Målsättningen är således att hitta en tidtabell som ger mindre förseningar än dagens, men det är också att dra ner antalet tidtabeller genom att välja ut de bästa, samt att minska modellens exekveringstid per simulering.

1.4 Definitioner

För att läsaren lättare ska kunna följa med i fortsättningen definierar vi här några begrepp som är vanlig förekommande.

- Regularitetsbrott – Om ett tåg är mer än 2,5 minut försenat vid en viss station innebär det ett regularitetsbrott.

- Regularitet – Andelen avgångar som inte begår regularitetsbrott. Om 10 tåg har stannat och avgått från en station under en period, och 3 av dessa har begått regularitetsbrott, så har stationen en regularitet på 70 procent.

- Rusningstid – Uppstår mellan 06.00-10.00 och innebär mer trafik än vanligt. Tågen avgår också ungefär dubbelt så ofta under den här tiden.

- Minsta körtid – Den tid det minst tar för ett tåg att färdas från station A till B.

- Minsta stopptid – Den tid som det minst tar för tåget att öppna dörrarna och stänga dörrarna igen. Denna tid är 16 sekunder.

- Tidtabell – Den tabell som lokförarna använder för att veta när tågen ska avgå och ankomma. Den tillåter endast avgång vid jämna halvminuter. Den vanliga tidtabellen som används för passagerarna tillåter endast avgång vid hela minuter. Vi har därför valt att använda lokförarnas tidtabell då den blir mer precis.

- Bufferttid – Om en tidtabell endast varit sammansatt av den tid som är absolut nödvändig för att ta sig från station A till station B, det vill säga minsta körtid och minsta stopptid, så hade tågen nästan aldrig kommit i tid. Bufferttiden används för att tågen ska få lite spelrum. Bufferttiden ges således av:

  $$Buffertid = Tidtabellerad\ tid - minsta\ körtid - minsta\ stopptid$$

  Till exempel blir bufferttiden för Ballerup följande:

  Tåget avgår 06.43 från Herlev, som är stationen före och kommer fram 06.48 i Ballerup. Minsta körtiden är 4 minuter och 28 sekunder, och stopptiden är som vanligt 16 sekunder.

  $$Bufferttid = (48 - 43) - 4\ min. 28\ sek. - 16\ sek. = 16\ sek.$$

Tåget skulle till exempel också kunna avgå 06.48’30, eller 06.49, och då skulle bufferttiden blivit 46 sekunder, respektive 76 sekunder. Den sammanlagda bufferttiden för en hel linje får max vara 14 procent av den sammanlagda minsta körtiden. Då bufferttiden är det enda som inte är konstant i tidtabellen kommer det att vara denna vi varierar när vi säger att vi testar olika tidtabeller.

- Avrundningsbufferttid – Den bufferttid som är absolut nödvändig för att tåget ska kunna avgå vid ett halvt antal minuter. Även om ett tågs teoretiska ankomsttid är 07.35’22, så måste tiden avrundas till 07.35’30. Den nödvändiga bufferttiden är då 8 sekunder. Den totala bufferttiden vid en station är avrundningsbufferttiden adderat med godtyckligt antal halvminuter. Såsom 8 sekunder, 38 sekunder, 1 minut och 8 sekunder, och så vidare.

- Mätstationer – Är de stationer som föregående examensarbetare bestämt sig att utgå från. Varje linje består av cirka 8 mätstationer, och deras bufferttid har ansetts som konstanta. Tiden man använt kommer från den befintliga tidtabellen. Optimeringen har sedan skett på de stationer som finns mellan mätstationerna.

1.5 Avgränsningar och förenklingar

Modellen körs endast under de fyra timmar som man har så kallad rusningstrafik. Den är nämligen mest kritisk och mest intressant att titta på.

De stokastiska indata som används för stationerna är samma i både norr- och södergående riktning, samt att de gäller endast för på och avstigning och inte för körtiden, som i modellen alltid är samma.

Modellen är anpassad för att endast simulera normal trafik. Den tar ej hänsyn till tåghaveri eller andra omständigheter som kan anses vara utöver det normala. DSB anser att detta händer så sällan att det inte ska påverka normaldriften. Dessutom hade det blivit för omfattande att lägga in sådana händelser i modellen.

När tågen avverkat sin linje i en riktning ska de i verkligheten köra tillbaka i motsatt riktning. För att underlätta optimeringen och minimera simuleringstiden har vi valt att göra varje riktning självständig, det vill säga att när tåget kört klart i en riktning börjar den om från början i samma riktning. Det har inneburit lite annorlunda resultat för de

enskilda linjerna, vilket kan ses i Tabell 1, men då det också blivit en rejäl förbättring för modellens tidsprestanda så har skillnaderna ansetts acceptabla. Det innebär också att en linje får ta ansvar för sina egna förseningar, istället för att låta den andra riktningen ta över problemen. Vilket med största sannolikhet faktiskt ger ett mer sanningsenligt resultat.

Linje	Procentuell skillnad
Linje E(s)	-0,64 %
Linje B(s)	-1,30 %
Linje A(s)	-0,05 %
Linje Bx(s)	0,51 %
Linje H(s)	-0,43 %
Linje C(s)	-1,06 %
Linje E(n)	-1,03 %
Linje B(n)	0,21 %
Linje A(n)	-0,18 %
Linje Bx(n)	0,40 %
Linje H(n)	0,21 %
Linje C(n)	-0,48 %

Tabell 1: Procentuell skillnad vid simulering av samma indata för två olika modeller. En modell med de vanliga inställningarna, som sedan jämförts med en modell där tågen inte byter riktning.

DSB:s trafikledning kan egentligen avlysa tåg när de är mycket försenade. Detta är för att undvika följdförseningar, men i modellen finns inte den funktionen.

Om än önskvärt så vore det praktiskt omöjligt att testa alla kombinationer av tider för tidtabellen. Därför har vi valt att endast testa de tider vi på goda grunder anser har möjlighet att bli optimala. Dagens tidtabell har således använts som grund, eftersom det ansetts orimligt att de värdena skulle vara helt fel.

1.6 Användningsområde

Vi har som förhoppning att modellen av S-tågnätet och den utvecklade optimeringsheuristiken kommer att kunna användas som hjälpmedel vid framtagning av

nya tidtabeller. Eventuellt kan även optimeringskonceptet användas inom andra områden där man inte kan simulera alla tänkbara kombinationer.

1.7 Målgrupp

Målgruppen för detta arbete är DSB och analysgruppen för S-tåg, speciellt vår handledare Stefan Vidgren. Även övriga studenter på Lunds Tekniska Högskola som finner ämnet intressant, ser vi som en tänkbar målgrupp.

1.8 Uppsatsens struktur

Kapitel 1 - Inledning: Inledningen är till för att orientera läsaren i vad arbetet innehåller. I detta inledande kapitel ges en kort bakgrund av uppdragsgivaren samt uppgiften. Problematisering, syfte och avgränsningar till uppgiften presenteras. Delkapitlet med definitioner är viktigt att beakta.

Kapitel 2 - Metod: I detta kapitel behandlas metoder och ansatser som har för avsikt att utgöra ett ramverk för arbetet. Avgränsningar, validitet och reliabilitet kommenteras också.

Kapitel 3 - Teori och kort om dess applicerande: För att kunna genomföra examensarbetet har vi varit tvungna att ta hjälp av befintlig teori. Den presenteras i detta kapitel med en kort beskrivning i slutet om hur den har applicerats i arbetet.

Kapitel 4 - Idébeskrivning: Den konceptuella idén presenteras utan för stor detaljrikedom. Från verklighet till modell, och sedan simulering, och hantering av data.

Kapitel 5 - Modellbeskrivning: Modellen vi använt för simulering är ett resultat av ett annat examensarbete av Patrik Bjärkeson och Björn Johnsson. Vi vill i detta kapitel kortfattat beskriva modellen och även redogöra för de förändringar vi infört.

Kapitel 6 - Utförande: Här redogörs för det praktiska tillvägagångssättet för examensarbetet.

Kapitel 7 - Resultat: Resultaten av examensarbetet presenteras. Både eventuella förbättringar av regulariteten och hur mycket vi fick ner antalet simuleringar till.

Kapitel 8 - Slutsats och diskussion: Resultat och modellen diskuteras, det presenteras även idéer och förslag på framtida arbete.

Kapitel 9 - Referenser: De referenser som använts presenteras.

2 Metod

I detta kapitel behandlas metoder och ansatser som har för avsikt att utgöra ett ramverk för arbetet. Avgränsningar, validitet och reliabilitet kommenteras också.

2.1 Ansats[3]

Ansatsen beskriver vilket förhållande författaren väljer att ha mellan befintlig teori och den empiri som arbetet utformas efter. En ansats kan vara:

- *Induktiv* – Utgår från empirin, och utifrån den görs försök att finna nya teorier.
- *Deduktiv* – Utgår från befintliga teorier och utifrån dessa samt med hjälp av empiri görs försök att finna nya underordnade teorier.
- *Abduktiv* – Är ett mellanting mellan induktiv och deduktiv ansats, och växlar mellan empiri och teori. Benämns också som ”den gyllene medelvägen”.

2.1.1 Vald ansats

Då vi inte har någon befintlig teori som vi utgår från, utan snarare startar med empirin för att hitta en optimal lösning kan vi anse att vår ansats är induktiv. Vi tar dock hjälp av teorin under vägens gång.

2.2 Metodik[4]

Metodik är arbetssättet man väljer att utföra examensarbetet på. Vald metod skall snarare ses som ett ramverk än att i detalj föreskriva vad som skall göras samt icke göras. De olika typerna av arbete kan ha som huvudsyfte att vara:

- *Beskrivande* – Beskriver hur något fungerar eller utförs.
- *Utforskande* – Djupstudier, som har till syfte att förstå hur något fungerar eller utförs.

[3] Att genomföra examensarbete Höst, Regnell, Runeson. 2006
[4] Att genomföra examensarbete Höst, Regnell, Runeson. 2006

- *Förklarande* – Söker orsakssamband för hur något fungerar eller utförs.

- *Problemlösande* – Lösning av ett problem som har identifierats.

Data som behandlas i arbetet kan vara av olika karaktär:

- *Kvantitativa data - S*ådant som kan räknas eller klassificeras: antal, andel, längd etcetera.

- *Kvalitativa data* - Sådant som är av beskrivande form. Intervjuer och dylikt, ofta rik på detaljer.

Arbetet kan mycket väl innehålla både kvantitativa och kvalitativa data. För att analysera kvantitativ data kan statistisk analys användas. För att analysera kvalitativ data krävs ofta att man kategoriserar och sorterar.

En metodik kan ha olika design som begränsar hur arbetet får utvecklas med tiden:

- *Fix natur* - Arbetet är definierat innan man påbörjat själva genomförandet

- *Flexibel natur* - Arbetet går att ändra på under tiden man genomför det.

Det finns huvudsakligen fyra olika metoder som används vid författning av ett examensarbete inom tillämpade vetenskapsområden. Dessa är:[5]

- *Kartläggning* – Sammanställning och beskrivning av nuläget för det studerade objektet och fenomenet.

- *Fallstudie* – Djupgående studium av ett eller flera fall där man försöker påverka det studerande objektet så lite som möjligt.

- *Experiment* – Jämförande analys av två eller flera alternativ, där man försöker isolera ett fåtal faktorer och manipulera ett av dem.

- *Aktionsforskning* – En noggrant övervakad och dokumenterad studie av en aktivitet som syftar till att lösa ett problem.

---

[5] Att genomföra examensarbete Höst, Regnell, Runeson. 2006 sidan 30

I Tabell 2 förtydligas vilka egenskaper de olika metoderna har.

Metod	Huvudsyfte	Data	Design
Kartläggning	Beskrivande	Kvantitativ	Fix
Fallstudie	Utforskande	Kvalitativ	Flexibel
Experiment	Förklarande	Kvantitativ	Fix
Aktionsforskning	Problemlösande	Kvalitativ	Flexibel

Tabell 2: Sammanfattning av de vanligaste metoderna och dess syfte, data och design[6].

2.2.1 Vald metodik

Det övergripande syftet med uppgiften var att lösa ett problem för vår uppdragsgivare. Då vi utför jämförande analys av antalet regularitetsbrott vid olika val av en isolerad faktor (tidtabellsinställningar) blir den valda metoden för att lösa problemet en experimentstudie, men med vissa modifikationer. Huvudsyftet är snarare av problemlösande natur än förklarande. Data vi samlar in är dock helt klart kvantitativ, eftersom det endast handlar om antalet regularitetsbrott för varje linje. Designen får anses fix, då ändringar i efterhand inte är möjliga. Det skulle till exempel inte gå att först simulera en linje med en viss modell, och sen göra uppdateringar i modellen för nästa linje. Vi har i och för sig varit tvungna att utföra en del ändringar på vägen, men då har vi också börjat om från början på hela optimeringen. Reproducerbarhet innebär möjligheten att upprepa ett experiment, och är en viktig del när man utför sådana. I vårt fall är reproducerbarheten för varje enskild körning inget problem då inte förutsättningarna ändras alls. Endast indata förändras. Reproducerbarheten för hela optimeringen är en knivigare fråga. Hade två andra examensarbetare kommit fram till samma lösning som vi, eller hade de funnit en annan optimal lösning? Hade vi testat alla möjliga värden hade det säkert varit så, men eftersom vi endast testar de varianter på värden som vi finner ha goda möjligheter att bli optimala, så kan resultatet avvika något. Det blir ju då fråga om en något subjektivare bedömning.

[6] Att genomföra examensarbete Höst, Regnell, Runeson. 2006. Sida 43

2.3 Mätteknik[7]

Det är givetvis viktigt att vi mäter det vill mäta och att det vi mäter mäts korrekt. Detta problem benämns vanligtvis som validitet respektive reliabilitet och behandlas i nästa kapitel. Lite mer allmänt om mätningar kan sägas att de kan vara direkta eller indirekta. Om man vill mäta hur långt något är, mäter man med en linjal direkt på objektet och läser av värdet. Indirekta mätningar utförs om det riktiga objektet är svårt att mäta. Då mäter man på något som kan användas som ett substitut och är enklare att mäta. I vårt fall har vi indirekta mätningar då vi vill mäta hur regulariteten påverkas i verkligheten, men blir tvungna att mäta av resultatet i vår modell av verkligheten. Förhoppningsvis kommer ett bra värde i modellen också bli ett bra värde i verkligheten, men det är inte lika säkert som när man mäter direkt på objektet.

Mätfel brukar delas in i tre olika sorters fel, och kan symboliseras på en darttavla där centrum är det sanna värdet[8]:

- Grova fel (G) - Rejält avvikande resultat, som brukar uppstå på grund av fel i avläsning, eller i protokollförandet. Mänskliga misstag helt enkelt.
- Systematiska fel (S) - Fel som uppstår vid varje mätning. Brukar orsakas av någon felinställning, eller konstant störning.
- Tillfälliga fel (T) - Slumpmässiga variationer kring väntevärdet. Vill vi mäta PH-värdet i en sjö kommer vi förmodligen inte få exakt samma värde varje gång. Medelvärdet av 10 mätningar kan däremot bli bättre.

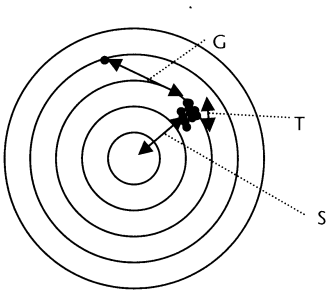


Bild 1: Mätningar i form av en darttavla. Grova fel (G), systematiska fel (S), tillfälliga fel (T).

[7] Att genomföra examensarbete Höst, Regnell, Runeson. 2006
[8] Att genomföra examensarbete Höst, Regnell, Runeson. 2006 sidan 95

Grova fel i vårt examensarbete är nästan obefintliga, då vi låter en dator utföra mätningarna och även protokollförandet. Det skulle kunna finnas fel i koden, men då blir det snarare ett sytematiskt fel, eftersom samma fel uppkommer hela tiden.

Då vi mäter resultatet på en modell av verkligheten kommer vi förmodligen att få ett systematiskt fel, som gör att vi hela tiden hamnar lite fel i våra mätningar. Eftersom det rör sig om jämförelser och relativa förändringar kommer detta förhoppningsvis inte bli något problem i slutändan.

Mätningarna har ganska stora slumpmässiga variationer och skiljer sig mycket från gång till gång. Det blir därför viktigt för oss att göra många mätningar av samma indata.

2.4 Validitet

Validitet betyder att man mäter det man ska mäta. Det avser kopplingen mellan det objekt man undersöker och det man faktiskt mäter. Om man vill undersöka hur långa människorna i en viss population är, vore det kanske dumt att mäta hur mycket personerna väger.

Vi önskar mäta regulariteten i verkligheten, men det är inte praktiskt möjligt. Därför använder vi oss av en modell av verkligheten. Ju bättre den är desto bättre validitet får vi, eftersom vi då mäter rätt sak.

Då vi i vissa fall vill ha ett mätresultat som inte kommer direkt ur modellen får vi skriva en algoritm som ser till att mäta rätt saker. Då blir också resultatet beroende av denna algoritm. Ett valideringsproblem vi haft är att vi endast mäter regulariteten på ett visst antal mätstationer, och inte på samtliga. Annars skulle det inte vara möjligt att optimera systemet. Vi har dock goda belägg för att mätningen av just dessa stationer kommer ge ett sanningsenligt resultat för hela systemet, då vi har testat detta mot kända data. Se Tabell 3. Resultatet för mätstationerna är där i stort sett samma som för om man kört för alla stationer. Man skulle kunna säga att vi mäter storleken på skon, då vi egentligen vill mäta storleken på foten. Resultatet kommer då med största sannolikhet bli ett bra värde.

Linje	Skillnad	Intervall
Linje E(s)	0,32%	-0,84% - 1,48%
Linje B(s)	0,30%	-1,21% - 1,8%

Linje A(s)	0,51%	-0,48% - 1,49%
Linje Bx(s)	2,90%	1,41% - 4,39%
Linje H(s)	-1,93%	-4,36% - 0,5%
Linje C(s)	-2,56%	-4,38% - -0,74%

Linje E(n)	-4,79%	-5,91% - -3,67%
Linje B(n)	2,87%	1,53% - 4,20%
Linje A(n)	-2,93%	-4,23% - -1,63%
Linje Bx(n)	12,14%	9,88% - 14,39%
Linje H(n)	-4,85%	-6,92% - -2,79%
Linje C(n)	7,49%	5,88% - 9,10%

Tabell 3: Skillnaden mellan att mäta på vanliga stationer jämfört med att bara mäta på mätstationer.

Vi använder indata för av- och påstigning från DSB:s databaser, men tyvärr är den föråldrad och har brister såsom att det är samma tider för södergående som norrgående riktning. Detta kommer att påverka resultatet och göra det mindre trovärdigt. Däremot är algoritmen så pass universell att det går att ändra indata i efterhand och göra en ny optimering med mer korrekt resultat.

Vår uppdragsgivare har instruerat oss att mäta antalet regularitetsbrott vid varje mätning, detta för att minska antalet tåg som är försenade. Vi har ingen ambition att ändra deras definition på vad som anses vara ett tåg som kommer i tid, men man skulle kunna tänka sig att den ackumulerade förseningen hade varit ett mer korrekt värde att mäta, och optimera efter. I alla fall om målet är fler tåg i tid. Antalet regularitetsbrott har tidigare varit bötesbelagt och vi tror det är därför som man fortfarande använder detta för att mäta resultatet internt.

2.5 Reliabilitet

Reliabilitet står för att det vi önskar mäta blir mätt på ett korrekt sätt, och att vi har en metod för att avvärja fel uppkomna på grund av slumpmässiga variationer. Om vi vill mäta en persons vikt, och behöver ett resultat som är väldigt noggrant, borde vi inte använda oss av en våg som är avsedd att mäta vikten på till exempel ett fartyg. Då den förmodligen inte skulle ge tillräckligt noggranna resultat. Vi bör också mäta flera gånger för att undvika slumpmässiga variationer.

I vårt fall är det väldigt enkelt att mäta. Det är inga krångliga experiment som ska mätas av en människa, och det är inga slumpmässiga variationer för mätinstrumentet. En dator mäter våra resultat och behandlar dessa. Väldigt små risker finns därför för mänskliga fel. Det skulle kunna finnas slarvfel i programmeringskoden, men detta har vi gjort vårt yttersta för att motverka, bland annat genom att testköra koden mot indata som ger kända utdata.

De slumpmässiga variationerna på våra utdata är ganska markanta, men det åtgärdar vi genom att simulera samma indata ett tillräckligt antal gånger.

2.6 Simulering

Definitionen av simulering är enligt Shannon R.E. ”System Simulation: the art and science”, 1975, fritt översatt:

”Simulering är processen för att designa en modell av ett verkligt system, och genomföra experiment med denna modell med ändamålet att antingen förstå beteendet av systemet, eller att utvärdera olika strategier för styrning av systemet”

Simulering är ett kraftfullt verktyg när experiment i verkligheten blir för kostsamma. Tiden det tar att utföra varje experiment kan kraftigt minskas då man använder sig av simulering, och reproducerbarheten för varje experiment blir perfekt. Det är praktiskt mycket enkelt att utföra ett experiment med exakt samma förutsättningar som föregående gång, vilket är väldigt svårt i verkligheten då yttre icke mätbar påverkan ofta stör. Det är även möjligt att simulera händelser som i vanliga fall skulle innebära en stor risk att genomföra.

I vårt fall är en simulering det enda tänkbara alternativet då testkörningar i verkligheten hade varit praktiskt ogenomförbart. Tänk er själva hur ni hade reagerat om tidtabellen för ert tåg hade ändrats varje dag, i avsikt att hitta en optimal lösning. Det hade inte tjänat sitt syfte.

Första steget vid en simulering är att skapa själva modellen. Den kan utformas på lite olika sätt. Generellt kan man säga att den kan vara:

- **Deterministisk eller stokastisk** - I en deterministisk modell kan utdata beräknas direkt när indata är kända såsom en matematisk formel. Utdata är alltså en direkt funktion av indata. I en stokastisk modell är utdata även beroende av slumpmässiga händelser. Utdata kan därför inte direkt beräknas.

- **Statisk eller dynamisk** – I en statisk modell tar man inte hänsyn till tiden. Det gör man i en dynamisk.

- **Diskret eller kontinuerlig** – I en kontinuerlig modell förändras systemvärdena med tiden, medan i en diskret modell förändras systemvärdena endast vid vissa diskreta tidpunkter.

Tidsaspekten för en modell som är diskret kan behandlas på två olika sätt. Antingen *"next event"*, eller *"time slice"*. *"Time slice"* innebär att ett tidtagarur med givet tidsintervall hela tiden räknar framåt ett tidsintervall åt gången. Händer det något i detta tidsintervall registreras detta och utförs. *"Next event"* räknar bara fram tiden då det verkligen händer något, och hoppar över den tid som finns mellan händelserna.

Modellen vi använder är stokastisk, dynamisk, och diskret. Den använder sig av *"next event"* metoden.

2.7 Använda programvaror

För att kunna genomföra examensarbetet har vi använt oss av följande verktyg.

2.7.1 Extend[9]

Extend är ett simuleringsprogram som kan bygga upp grafiska modeller av verkligheten. När modellen är klar kan man testa olika inställningar för viktiga parametrar, och förhoppningsvis hitta de inställningar som är optimala. Extend innehåller många smarta hjälpmedel som tillåter användaren att på ett enkelt sätt bygga modeller och se var köer och problem uppstår.

Till exempel kan man bygga en modell över ett sjukhus där man varje dag utför ett antal aktiviteter. Aktiviteterna tar en viss tid att utföra, och man har tillgång till ett bestämt antal vårdbiträden som kan utföra aktiviteterna. Modellen kan då till exempel visa hur långa köer som uppstår, var de uppstår, och förhoppningsvis peka på vad som behöver förbättras. Kanske bör det införas fler vårdbiträden? I programmet ändras då enkelt hur

[9] http://www.extendsim.com/prods_overview.html 2009-05-05

många vårdbiträden som finns på plats, och en ny simulering genomförs. Ett bättre resultat pekar då på att sjukhuset helt enkelt borde anställa fler.

Extend kan simulera både diskreta och kontinuerliga händelser, och eftersom programmet byggs upp grafiskt och inte genom kodskrivning, blir det enkelt att bygga modeller och felsöka. Enkelheten har dock ett pris, och det är att programmet inte blir lika flexibelt som ett program där man använder sig av kodskrivning samt att simuleringstiden kan bli lång för komplexa modeller. Det går även att koppla Extend till utomstående program, till exempel Excel, för att underlätta beräkningar.

Extend innehåller även en optimeringsfunktion, som har för avsikt att allokera tillgängliga resurser på ett optimalt sätt. Denna har vi inte använt oss av då den inte är tillräckligt kraftfull för det stora system som S-tåg nätverket är, samt att tidigare examensarbetare redan påbörjat optimeringen utan denna funktion.

I vårt examensarbete har Extend använts flitigt, men mest som ett mätinstrument. Modellen av tågnätet har byggts upp i Extend, men till största delen är det Patrik Bjärkeson och Björn Johnsson som konstruerat den. Vi har dock gjort en del förbättringar samt tillverkat en manual för hela modellen, se bilaga vilket gjort att vi är väl insatta i hur den fungerar.

2.7.2 Excel[10]

Excel ingår i Microsoft Office och är ett program som lagrar och visualiserar data i ett kalkylblad. Det kan användas till beräkning och sortering av data på olika sätt. Excel har många inbyggda funktioner som underlättar beräkning, till exempel finns färdiga funktioner för att beräkna medelvärde och standardavvikelse. Med hjälp av programmets grafiska verktyg kan Excel presentera data på en mängd sätt, till exempel genom grafer och pivottabeller.

Excel har varit ett utmärkt verktyg för oss då vi enkelt kunnat lagra och visualisera olika inställningar för tågen, så som bufferttider och minsta körtid. Det har även fungerat som sorteringsverktyg för de stora mängder resultat som genererats under de olika körningarna.

[10] http://office.microsoft.com/sv-se/excel/default.aspx 2009-05-05

2.7.3 Visual Basic for Applications[11]

Visual Basic for Applications, VBA, är ett programmeringsverktyg som används till Microsofts applikationer, så som Excel och Word. Programmet använder som namnet antyder sig av ett liknande språk som Visual Basic men har även extra verktyg från de applikationer som det underliggande programmet innehar. Till exempel kan du anropa Excels metod för att beräkna medelvärde i VBA för Excel. Med hjälp av VBA kan användaren konstruera makron som möjliggör snabb datoriserad, och automatiserad hantering av data istället för att behöva göra det manuellt. Det gör helt enkelt Excel kraftfullare, och möjliggör mer detaljerad programmering än Excels standardfunktioner.

För vår del har allt beräkningsarbete skett via VBA. Då vi har stora mängder data som ska användas och kopieras vid varje simulering hade det tagit alldeles för lång tid att göra detta manuellt. Risken för mänskliga fel hade också blivit överhängande. VBA kan anropa andra program, till exempel Extend, och på så sätt automatisera även simuleringsprocessen.

2.8 Utvecklingsmetodik för programvara

Det här examensarbetet har krävt att vi utvecklat en del programvara och beräkningsapplikationer, till största delen i Visual Basic for Applications, samt Extend. Det ska dock nämnas att det inte är i ovan nämnda programs grundfunktion vi har ändrat, utan i modellen i Extend, samt beräkningsapplikationerna i Visual Basic. Till vår hjälp när vi utvecklat dessa har vi haft *Royce Waterfall approach*. Den kännetecknas av att man genomför ett steg komplett för att sedan gå vidare till nästa. Stegen är:[12]

- *Specifikation* – mjukvarans egenskaper och begränsningar specificeras noggrant.
- *Design och implementering* – huvudstrukturen av programvaran designas och komponenterna till programvaran identifieras. Dessa implementeras sedan med hjälp av ett programmeringsspråk

[11] http://msdn.microsoft.com/en-us/isv/bb190538.aspx 2009-05-05
[12] Software process models, Sommerville Ian, ACM Computing Surveys, 1996

- *Integration och testning* – individuellt utvecklade moduler integreras och en valideringsprocedur genomförs.

- *Användning och underhåll* – programvaran levereras till sin användare, programmet uppdateras i takt med att fel upptäcks och nya behov uppstår.

När vi utvecklade programvaran började vid med att skissera en specifikation och enkelt tänka ut och designa vad programmet skulle producera och beräkna. De flesta av de applikationer vi utvecklade hade en simuleringsdel i sig, och då var vi tvungna att göra ett överslag hur mycket tid det fick ta i anspråk, och på så sätt begränsa programmets omfång.

När väl den övergripande delen var gjord började vi skriva den kod som behövdes till applikationen. När koden var implementerad testkörde vi modulen innan vi integrerade den med andra applikationer och programvaror. Ett enkelt test var ett så kallat *"sanity-test"*, där programmet fick göra mycket enkla beräkningar där resultatet var känt. Fick vi samma resultat som det kända kunde vi anse att programmet var validerat. Vi byggde även enklare testapplikationer som kontrollerade att de numeriska värden vi använde oss av var korrekta. Efter testning av delapplikationen satte vi ihop den med simuleringsprogramvaran och genomförde ytterligare testkörningar för att kontrollera att programmet fungerade. Testkörningarna var av ett mycket enkelt slag där vi lät programmet göra beräkningar som vi samtidigt gjorde manuellt.

Mycket av den utvecklade programvaran i vårt examensarbete är dubblering av programvaran vi utvecklat. Till exempel är optimeringsapplikationen för de norrgående linjerna en spegelbild av de södergående linjerna, samt att linjernas optimeringsapplikationer bara har små förändringar mot varandra, principen är den samma för varje linje. Detta gjorde att större delen av utvecklingens gång kunde börja vid punkt tre, integration och testning, för de andra stegen var redan avklarade.

Slutligen när vi använde programmet upptäckte vi ibland större fel, detta gjorde att vi lade till fler säkerhetsapplikationer och justerade programvaran. Kraven på programvaran ändrade sig också över tiden och applikationerna fick justeras för att passa in. Till exempel bantade vi ner och tog bort vissa beräkningsalgoritmer för att minska exekveringstiden för programmet.

2.9 Källkritik

Indata för på och avstigning har inte varit helt uppdaterade som nämnts under reliabiliteten, och vi är medvetna om att det påverkar resultatet. Vår uppdragsgivare har flaggat för att det ska komma uppdaterade indata inom kort, dock för sent för att vi ska kunna använda det.

Källorna till litteraturstudierna får anses relativt säkra då det nästan alltid är vedertagna teorier från etablerade läroböcker. Användning av artiklar som ligger i forskningens framkant är mycket begränsad. Arbetet har mer handlat om vår problemlösande förmåga.

3 Teori och kort om dess applicerande

För att kunna genomföra examensarbetet har vi varit tvungna att ta hjälp av befintlig teori. Den presenteras här nedan med en liten beskrivning i slutet om hur den har applicerats i arbetet.

3.1 Statistikteori

Då examensarbetet har behandlat en stor mängd stokastiska data från en modell, är kunskap om att behandla den nödvändig. Därför presenteras i kapitlet nedan de teorier inom statistik som vi använt oss av.

3.1.1 Konfidensintervall

Konfidensintervall används för att ange inom vilket intervall som ett väntevärde sannolikt befinner sig. Betrakta ett slumpmässigt stickprov

$$\boldsymbol{x} = (x_1, .., x_n)$$

från en fördelning som beror av den okända parametern θ. Det intervall I_θ som med sannolikheten $1 - \alpha$ sträcker sig över θ kallas konfidensintervall θ med konfidensgraden $1 - \alpha$. Punkterna som anger gränserna för intervallet kallas konfidensgränserna, och betecknas vanligtvis a_1 och a_2. Ett konfidensintervall kan skrivas som

$$P[a_1(\boldsymbol{X}) < \theta < a_2(\boldsymbol{X})] = 1 - \alpha$$

Oftast väljer man sannolikheten lika med 0,95, 0,99 eller 0,999 beroende på hur säker man vill vara, men det går att välja ett helt godtyckligt tal mellan noll och ett. Om man vill ha en risk på 0,05 att konfidensintervallet inte täcker över θ, väljer man $1 - \alpha = 0{,}95$.

Låt $x_1 \ldots x_n$ vara ett slumpmässigt stickprov från $N(\mu, \sigma)$. Antag också att standardavvikelsen σ är känd. Då gäller att det aritmetiska medelvärdet är fördelat $N(\mu, D)$, där $D = \frac{\sigma}{\sqrt{n}}$ och där n är antalet stickprov.

Ett tvåsidigt konfidensintervall kan då beräknas enligt följande:

$$I_{\mu=}\left(\mu-\frac{\lambda\alpha_{/2}\sigma}{\sqrt{n}},\mu+\frac{\lambda\alpha_{/2}\sigma}{\sqrt{n}}\right)$$

Här är $\lambda\alpha_{/2}$ kvantlien med den önskade signifikansnivån α.

Vi kommer att använda denna metod för att presentera de resultat vi får på de olika körningarna. Till exempel, *"Linje Es regularitet är idag mellan 91,1 och 92,3 procent med 95 procents säkerhet".*

3.1.2 Hypotestest vid normalfördelning[13]

Hypotestest används generellt då man vill testa och bestämma om en mängd värden ur en population är statistiskt signifikanta, eller endast beror på slumpen. Ett vanligt exempel på hypotestest är att man ställer man upp en nollhypotes, H_0, och jämför denna mot hypotesen H.

H_0: $\mu=\mu_0$

H: $\mu\neq\mu_0$

Sedan bestäms en signifikansnivå som anger felrisken för antagandet. Ju högre signifikansnivå desto säkrare blir antagandet, men det innebär också att det blir svårare att bevisa. Signifikansnivån innebär sannolikheten att förkasta nollhypotesen då den faktiskt är sann.

En testvariabel införs som benämns Z. Den normeras och ett avstånd från nollhypotesen beräknas enligt följande.

$$Z=\left|\frac{X-\mu}{\frac{\sigma}{\sqrt{n}}}\right|$$

X är den data vi vill testa mot μ.

[13] Sannolikhetsteori och statistikteori med tillämpningar, Gunnar Blom et al. 2005

μ är nollhypotesen. Det uttalade medelvärdet i populationen.

σ är den kända standardavvikelsen i populationen.

n är antalet, det vill säga storleken på populationen.

Är nu variabeln Z större än **|λ|**, som är det avstånd i standardavvikelser från nollhypotesen **μ,** för given signifikans så kan hypotesen förkastas. Värdet på **|λ|** kan ses i Tabell 4 nedan.

|λ|	1,96	2,58	3,29
α	0,05	0,01	0,001
Signifikans	95 %	99 %	99,9%

Tabell 4: λ för olika signifikans givet en normerad normalfördelning, N(0,1).

Vi använder oss av hypotestest när vi testar olika inställningar för tågtidtabellerna, detta för att minimera simuleringstiden. Egentligen borde samtliga inställningar testas cirka hundra gånger för att minska de slumpmässiga variationerna, men genom användning av hypotestest kan vi förkasta riktigt dåliga inställningar på ett tidigt stadium. Är en inställning signifikant sämre än den bästa är det inte nödvändigt att simulera den fler gånger och den kan därmed förkastas direkt.

3.1.3 Kombinatorik

Kombinatorik handlar om att räkna ut hur många sätt ett givet antal element kan arrangeras i. Det går till exempel med hjälp av kombinatorik att räkna ut hur många kombinationer av par i ess det finns redan vid given i poker, eller hur många olika kombinationer av koder det finns i ett kodlås med sex rullar märkta med siffrorna noll till nio.

Den enklaste satsen av kombinatorik kallas för **dragning med återläggning, och med hänsyn till ordning**. Med den kan det beräknas hur många kombinationer man kan dra av n element ur den givna mängden N. Varje gång ett element dras läggs det tillbaka och kan med samma sannolikhet dras igen. Antalet kombinationer blir då N^n. Exemplet ovan med kodlåset blir i så fall 10^6 om N symboliserar antalet siffror på rullarna det vill säga tio, och n symboliserar antalet rullar, det vill säga sex.

Nästa steg blir återigen **dragning med återläggning, men nu utan hänsyn till ordning**. Det vill säga att kombinationen 135789 är samma som 317985 då det inte spelar någon roll vilken inbördes position siffrorna har. Antalet kombinationer blir då ganska mycket färre. Närmare bestämt $\binom{N+n-1}{n}$ = $\binom{10+6-1}{6}$ = 5005 kombinationer.

Om inte elementet läggs tillbaka efter varje dragning, vilket innebär att om den första dragningen ger en etta, så finns ingen möjlighet för resterande dragningar att bli ett. Det kallas dragning utan återläggning. Om hänsyn tas till återläggning fås **dragning utan återläggning med hänsyn till ordning**. Då blir det $N(N-1)\cdot \ldots \cdot (N-n+1)$ olika kombinationer. Detta kallas även antalet permutationer av n element bland N. I fallet med kodlåset innebär det att samma siffra inte får förekomma mer än en gång. 123456 är alltså en giltig kombination, men inte 112345. $N(N-1)\cdot \ldots \cdot (N-n+1)$ = $10(10-1)(10-2)(10-3)(10-4)(10-5)$ = 151200 kombinationer

Givetvis finns även **dragning utan återläggning och utan hänsyn till ordning** vilket blir $\frac{N(N-1)\cdot \ldots \cdot (N-n+1)}{n!} = \binom{N}{n} = \binom{10}{6}$ = 210 kombinationer

I Tabell 5 nedan visas ett sammandrag av ovanstående.

	Med återläggning	Utan återläggning
Med hänsyn till ordning	N^n	$N(N-1)\cdot \ldots \cdot (N-n+1)$
Utan hänsyn till ordning	$\binom{N+n-1}{n}$	$\binom{N}{n}$

Tabell 5: Tabell över mängden kombinationer för olika fall

Nämnas bör även multiplikationsprincipen som innebär att om åtgärd 1 kan utföras på k sätt och åtgärd 2 kan utföras på n sätt finns det $k \cdot n$ kombinationer som åtgärderna kan utföras tillsammans på.

Vi behöver kombinatoriska kunskaper när vi beräknar antalet möjliga kombinationer av tidtabeller. Att testa samtliga skulle vara helt omöjligt, då antalet kombinationer vida överstiger antalet atomer i universum. Stora delar av examensarbetet har gått ut på att dra ner på antalet kombinationer till en rimlig nivå. Inför varje ny simulering har vi därför kontrollerat hur många kombinationer, och därmed simuleringar, vi reducerat bort, och därefter tagit beslut om huruvida vi har möjlighet att köra så många.

3.2 Hierarkisk planering[14]

Då ett problem är för stort för att lösa helt och hållet från grunden, kan man dela in det i mindre problem som löses efterhand. Detta är vanligt inom till exempel produktionsplanering då det hade varit praktiskt omöjligt att planera ett helt tillverkningssystem på en gång. Detta kan då underlättas med så kallad hierarkisk planering. Inom produktionsplanering kan planeringen delas in enligt en taxonomi som föreslogs 1965 av Anthony. Den översta nivån för planering är, *strategisk planering*. I den strategiska planeringen tas beslut på hög nivå i företaget. Besluten handlar om att tillgodose de mål som krävs för att tillfredställa externa krav. Besluten kan till exempel röra plats och storlek för produktionsanläggningar, anskaffande av utrustning med mera. Planering på denna nivå ska ha en bredd och lång planeringshorisont. Beslut på denna nivå är väldigt viktiga, för de anger inriktningen på verksamheten som påverkar företagets konkurrenskraft.

Nästa steg i planeringen kallas *taktisk planering*. På detta stadium fokuseras beslutsfattandet på hur de tillgängliga resurserna bäst kan användas. Det kan till exempel röra beslut angående arbetskraftsallokering och övertidsuttag eller styrning av lagernivåer. Det sista steget i Anthonys taxonomi kallas *verksamhetskontroll*. I denna del av planeringen fattas beslut som rör den dagliga driften. Det kan gälla beslut om batchstorlek och sekvensering och/eller orderstyrning.

Ett problem med hierarkisk planering är att risken för suboptimering är överhängande och det gäller för ledningen att se till så att denna risk minimeras. Detta kan göras genom att man visualiserar de beroendeförhållanden som råder mellan de olika avdelningarna inom organisationen.

Hierarkisk planering används som ovan nämnts oftast när det gäller produktions- eller verksamhetsplanering, men det finns de som hävdar att det kan användas även inom andra områden. Det har vi tagit fasta på och använt oss av metodiken att dela upp ett stort, i det närmaste olösligt problem i delproblem. Att dela in linjerna i olika stationer i olika typer kan ses som att dela upp tågnätet enligt Anthonys taxonomi. Det vi kallar

[14] S.C. Graves et al., Eds., Handbooks in OR & MS Vol. 4 1993

huvudmätstationer, motsvarar den strategiska planeringen. Det vi kallar mätstationer får symboliseras av taktisk planering. Slutligen får de vanliga stationerna ses som verksamhetskontroll. Om detta står det mer utförligt i Kapitel 4, som behandlar idébeskrivningen.

3.3 Optimering och linjär programmering[15]

Linjärprogrammering används för att lösa problem där man önskar optimera något, till exempel vinst genom att bäst allokera tillgängliga resurser. Att det heter linjär programmering beror på att de matematiska funktionerna i modellen behöver vara linjära, det finns således också olinjär programmering. Att det heter programmering syftar inte direkt till just datorprogrammering utan är mer synonymt med planering.

För att lösa ett linjärprogrammeringsproblem börjar man med att sälla upp ett antal parametrar:

Z = Värdet av det vi vill mäta, det kan till exempel vara vinst.

i = symboliserar en resurs (i=1,2 … m)

j = symboliserar en aktivet (j=1,2 … n)

x_j = Nivå för aktivitet j, till exempel tillverkar vi 5 stycken av produkt 1. Det är x vi vill variera för att hitta den optimala lösningen.

c_j = Bidrag från aktivitet j, till exempel produkt 1 ger en vinst på 8 kronor, samma mått som Z

b_i = Tillgängliga resurser från i, till exempel fabrik 1 kan borra 10000 hål per dag.

a_{ij} = Mängden av resurser hos i som konsumeras av aktiviteten j, till exempel produkt 1 kräver 2500 hål.

Då gäller det att maximera funktionen

$$Z = c_1x_1 + c_2x_2 + \ ... + \ c_{1n}x_n$$

[15] Introduction to Operations Research. Hillier and Lieberman 2001

Med bivillkoren

$$a_{11}x_1 + a_{12}x_2 + \ \dots \ + \ a_{1n}x_n \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ \dots \ + \ a_{2n}x_n \leq b_2$$

$$a_{m1}x_1 + a_{m2}x_2 + \ \dots \ + \ a_{mn}x_n \leq b_m$$

Naturligtvis finns det andra infallsvinklar, kanske vill man minimera Z eller bara hitta en lösning för den. Bivillkoren kan då också förändras och $\leq$ kan ersättas av $\geq$ eller $=$, samt att det kan finnas andra typer av bivillkor, till exempel att en aktivitet endast får stå för ett visst antal procent av den totala produktionen.

För vissa problem finns det en direkt analytisk lösning, medan det för andra problem krävs andra lösningsformer. Ett sätt kan vara att visualisera problemet grafiskt och där hitta den optimala lösningen. Ett annat sätt är att helt enkelt testa de möjliga kombinationerna och se vilken som ger bäst resultat. Detta kan vara svårt då mängden av en produkt inte är heltal, då antalet kombinationer snabbt blir väldigt många.

Vi har inte använt oss av linjärprogrammeringsmetodik för att hitta en optimal lösning, utan snarare för att dra ner på antalet kombinationer av tidtabeller. Samtliga kombinationer har räknats upp med hjälp av kombinatorik och en algoritm har sedan förkastat de som inte klarat av bivillkoren. Det är också av vikt att nämna att vi endast har använt oss av heltal när vi varierat tidtabellerna, vilket varit en förutsättning för att inte behöva testa oändligt antal kombinationer.

4 Idébeskrivning

Kapitlet behandlar övergripande hur vi tänkt lösa problemet, utan för stor detaljrikedom.

4.1 Tidtabellen

Grundproblemet är att finna den tidtabell som ger minst förseningar totalt för hela S-tågnätet. Istället för att mäta försening, har vi mätt antalet regularitetsbrott. En tidtabell består dock av lite olika saker, vilket krånglar till det hela. En minsta körtid, en minsta stopptid samt en bufferttid samtliga förklarade under definitioner i första kapitlet. Delas tidtabellen upp i dessa tre delar finner man att endast bufferttiden går att variera. Det innebär att bufferttiden direkt kan ersätta tidtabellen. Kommer vi fram till en optimerad bufferttid har vi även funnit en optimerad tidtabell, eftersom det bara är att omvandla bufferttiden till motsvarande tidtabell.

$$Bufferttid = Tidtabellerad\ tid - minsta\ körtid - minsta\ stopptid$$

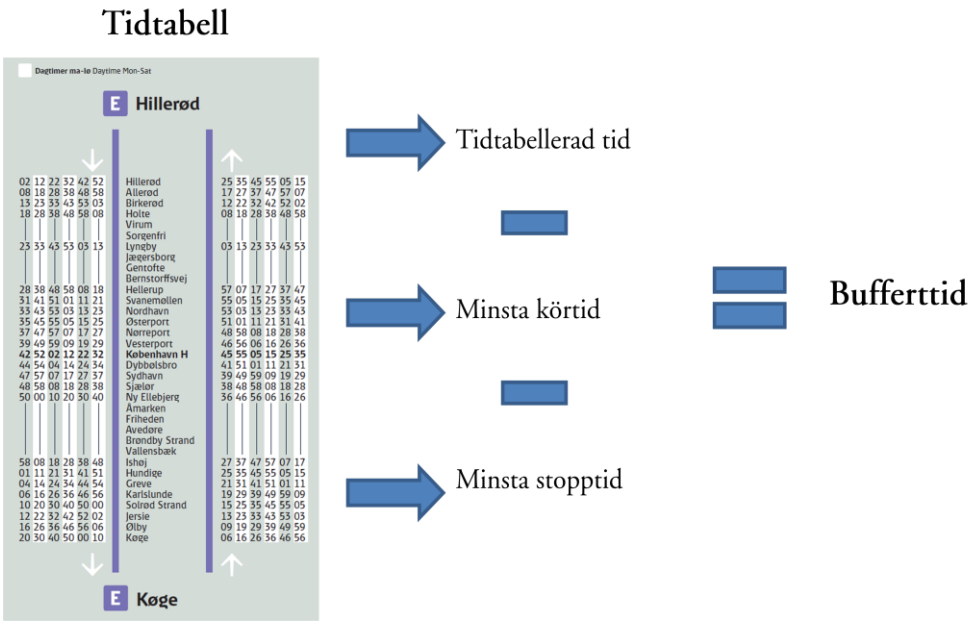


Bild 2: Tidtabellen kan omvandlas till bufferttid om man vet de konstanta värdena på minsta körtid och minsta stopptid.

I fortsättningen kommer vi därför att använda oss av bufferttiden istället för tidtabellen, då optimeringen utförs. Som nämnts tidigare under definitioner är det lokförarnas tidtabell vi optimerar.

4.2 Modellering, simulering, och hantering

Då det inte är möjligt att testa bufferttiderna i verkligheten har vi fått använda en **modell av verkligheten**, eller S-tågnätet. Modellen har tagits fram av Patrik Bjärkeson och Björn Johnsson, samt förbättrats något av oss. För att på ett smidigt sätt kunna behandla samtliga bufferttider, har vi använt oss av Excel. Olika varianter av bufferttid skickas från Excel till modellen i Extend. De **simuleras** i modellen och Extend svarar tillbaka till Excel med hur många regularitetsbrott som begåtts på respektive station och linje. Då det rör sig om många simuleringar och därmed många resultat behövs ett program som **hanterar** resultaten. Till detta har Visual Basic for Applications utnyttjats, som sparar undan resultaten för alla simuleringarna, behandlar dem statistiskt och presenterar en hanterbar sammanställning.

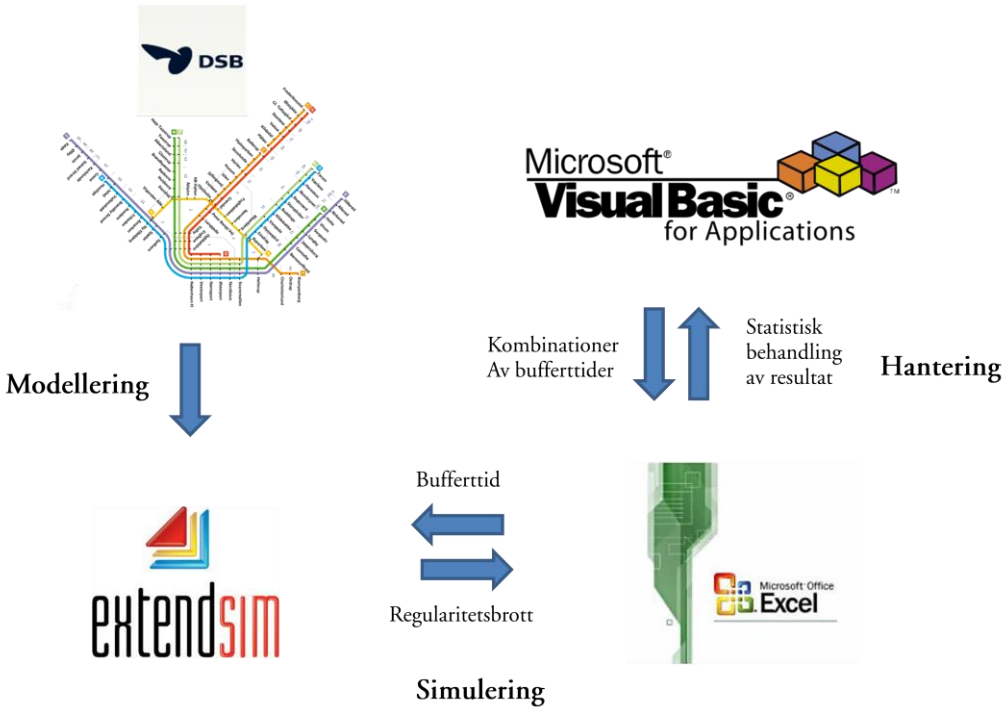


Bild 3: Från det verkliga S-tågnätet till färdigt resultat genom användning av programmen Extend, Excel, och Microsoft Visual Basic.

Visual Basic räknar även ut möjliga kombinationer av bufferttider och skickar dessa till Excel. Simuleringsheuristiken görs således i Visual Basic for Applications.

4.3 Uppdelning av stationer

För att möjliggöra den omfattande optimeringen var vi nödgade att dela upp den i tre delar. Samtliga stationer delades därför upp i *huvudmätstationer, mätstationer,* och *vanliga stationer.* Tanken var att först optimera huvudmätstationerna, sedan mätstationerna, och till sist de vanliga stationerna. Vid optimeringen av huvudmätstationerna adderas samtlig bufferttid till dessa stationer, och övriga reduceras till noll. Det samma görs för mätstationerna, men då hålls den redan optimerade bufferttiden för huvudmätstationerna konstant. Slutligen ska de vanliga stationerna optimeras och då hålls i sin tur mätstationerna konstanta.

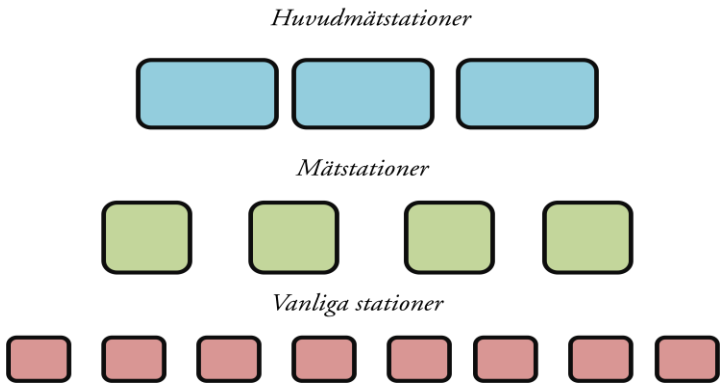


Bild 4: De tre olika stationstyperna i detta examensarbete.

4.3.1 Vanliga stationer och mätstationer - tidigare examensarbete

För att klargöra lite var vi har tagit vid efter de förra examensarbetarna, försöker vi här belysa deras insats. Efter att ha tagit fram och validerat en fungerande modell gjorde man även en optimering av den sista delen av systemets bufferttider, eller närmare bestämt de vanliga stationerna.

Varje linjes stationer fördelades mellan vanliga stationer och mätstationer och Bild 5 visar uppdelningen specifikt för en del av linje E. Fördelningen baserades på antagandet att, där linjerna slås ihop eller delas upp, bör man ha en mätstation. Detta är inget vi har ifrågasatt närmare, utan vi har accepterat mätstationerna och utgått från dem. Stationen Nordhavn

har dock vållat lite problem för våra beräkningar, och därför har vi tagit bort den som mätstation.

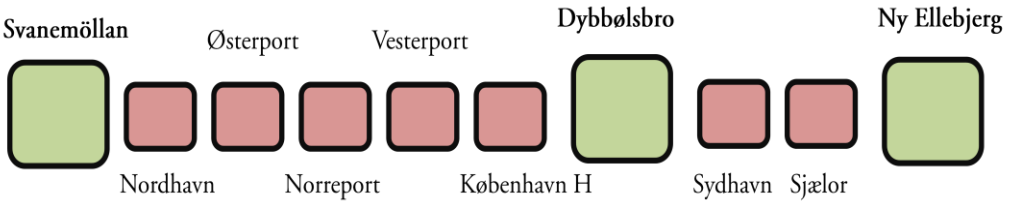


Bild 5: Uppdelning av en del av linje E, södergående riktning, i föregående examensarbete.

Vid optimeringen hölls bufferttiderna konstanta för mätstationerna. Man använde samma värde som dagens tidtabell, och endast bufferttiden på de vanliga stationerna ändrades och optimerades. Den sista delen i systemet kan alltså symboliseras av de vanliga stationerna, och det var denna del som föregående examensarbetare optimerade. Det som kvarstod var att optimera värdena för mätstationerna.

4.3.2 Centrala avsnittet

Det centrala avsnittet består av samtliga stationer mellan Svanemøllen och Dybbølsbro. Problematiken i detta område är att de olika linjerna måste ha exakt samma bufferttid. Detta är bestämt av DSB. Det innebär att om linje E har en bufferttid för Dybbølsbro som är 1,34 minuter, måste även linje A, B, C och F ha en bufferttid på 1,34 minuter. Detta gör att vi måste optimera det centrala avsnittet först, för att sen låsa värdena då vi optimerar övriga stationer. En ny uppdelning av stationerna blir därför nödvändig.

Bild 6: Det centrala avsnittet markerat med svart ring.

4.3.3 Huvudmätstationer

Begreppet huvudmätstationer införs för att simulera och optimera det centrala avsnittet. De används för att bestämma hur mycket tid som bör läggas före, under, samt efter det centrala avsnittet. Detta måste göras innan någon annan optimering kan påbörjas. Optimeringen för hela S-tågnätet börjar således med huvudmätstationerna, fortsätter med mätstationerna, och avslutas med de vanliga stationerna.

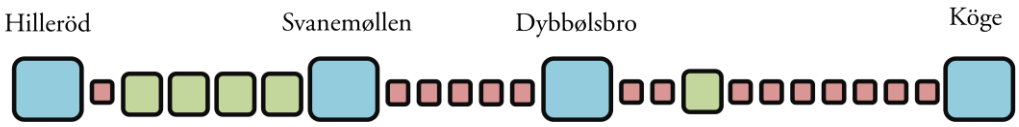


Bild 7: Uppdelning av linje E, södergående riktning, med namngivna huvudmätstationer.

Huvudmätstationerna utgör respektive linjes startstation, Svanemøllen, Dybbølsbro, och linjens slutstation. För linje E i södergående riktning blir det då: Hillerød, Svanemøllen, Dybbølsbro, och Køge. Här symboliserar Dybbølsbro bufferttiden under det centrala avsnittet, Svanemøllen symboliserar all bufferttid innan det centrala avsnittet, och Køge all bufferttid efter. Detta betyder att bufferttiden för Dybbølsbro, som representerade det centrala avsnittet, måste följa alla de restriktioner som beskrevs ovan.

Vid simulering sätts övriga stationers bufferttid till noll, och all borttagen bufferttid adderas till efterliggande huvudmätstation. Hillerød får därför inte någon bufferttid alls i ovanstående exempel. Användandet av huvudmätstationerna är något komplext och det kommer inte behandlas mer i detta avsnitt. Syftet är att hitta en bufferttid som är optimal för samtliga linjer i det centrala avsnittet, och sedan använda denna tid för optimeringen av mätstationerna.

4.4 Total simuleringstid

Om det finns ett behov av att göra 100 000 simuleringar, och varje simulering tar en minut, kommer den totala simuleringstiden att bli 100 000 minuter vilket är cirka 70 dygn. Varje sekunds förbättring som kan göras på tiden det tar att utföra en simulering ger alltså mer än ett dygns kortare total simuleringstid. Siffrorna stämmer ganska bra in på vårt examensarbete, och vi lade därför en del krut på att korta ner denna tid. Kanske inte för att korta ner den totala simuleringstiden utan snarare för att hinna med ett större antal simuleringar.

4.5 Möjliga kombinationer av bufferttider

Som vi tidigare nämnt hade det blivit alldeles för många kombinationer av bufferttider att simulera om vi valt att testa samtliga. Ett första viktigt antagande är, att vi endast testar halvminuter. Vi testar alltså inte alla tal mellan dessa halvminuter, då det hade inneburit oändligt många kombinationer i ordets sanna bemärkelse. Även om vi endast testar halvminuter kommer antalet kombinationer snabbt att bli för många. Ett grovt överslag ger följande:

Varje linje har i snitt 25 stationer, och vi antar att varje station endast kan variera sin tid på 15 olika sätt (från -1,5 minut till +6 minuter i steg om en halv minut). Totalt blir det då $25^{15} \approx 10^{21}$. Då vi har sex olika linjer i två olika riktningar och alla måste testas mot

varandra kommer siffran bli ännu högre, närmare bestämt $(10^{21})^{2\times6} \approx 10^{252}$. Detta är en ofantlig mängd kombinationer, och kan jämföras med den uppskattade mängden elementarpartiklar i universum som är 10^{80}. Hade vi kunnat använda all världens datorkraft i 10 000 år hade vi inte ens hunnit med att testa en mikroskopisk del.

Därför måste vi på något sätt dra ner på antalet kombinationer för att kunna genomföra simuleringsstudien.

- Separera de norrgående från de södergående linjerna, och behandla dem som om de inte påverkar varandra.

- Dela in alla stationer i tre olika sorter och optimera dem var för sig på ett hierarkiskt sätt.

- Utgå från den befintliga tidtabellen då den rimligtvis borde vara relativt bra redan nu. Istället för att alltid testa alla kombinationer av bufferttid från -1,5 minut till +6 minuter, testar vi då endast inom ett intervall på – 1 minut till + 1 minut jämfört med den befintliga tidtabellen.

- Låt den totala bufferttiden på varje linje vara konstant. Extrema värden på bufferttider försvinner då. Det kan till exempel inte vara hur stora bufferttider på en linje som helst, utan summan av bufferttiderna måste alltid bli samma. Se Tabell 6 i kapitel 5.1.

- Testa först ett mindre antal simuleringar på ett brett spektrum av kombinationer. Ett smalare intervall med de bästa kombinationerna ur ovanstående väljs sedan ut, och dessa simuleras mer ingående.

Efter dessa modifieringar är antalet kombinationer nere på rimliga nivåer. Närmare bestämt cirka 2000. Varje kombination måste simuleras 50 gånger för att de slumpmässiga variationerna ska försvinna. Därför blir då det totala antalet simuleringar cirka 100 000. Det bestämdes i kapitel 4.4 att detta skulle vara en rimlig mängd, eftersom tidsåtgången skulle bli cirka 70 dygn. Då vi faktiskt reducerat antalet kombinationer till ett ofantligt mycket mindre antal än den totala mängden, blir ju sannolikheten att hitta en kombination som är bättre än dagens tidtabell också mycket mindre. Förhoppningsvis har vi dock endast tagit bort kombinationer som ändå inte haft någon större chans att bli optimala. En stor del av examensarbetet har därför varit, att på goda grunder välja ut kombinationer som har en rimlig chans att bli bra.

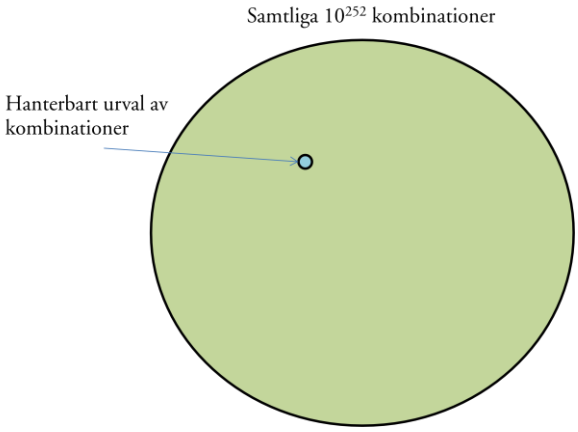


Bild 8: Det har endast funnits tid att simulera ett ofantligt litet antal kombinationer av den totala mängden. Detta minskar våra chanser att faktiskt hitta en optimal kombination.

4.6 Viktning av resultatet

Som vi redan poängterat i kapitel 2.4 om validitet, är det viktigt att mäta det vi verkligen vill mäta. Då vi ofta ändrar fokus på vad i tågnätet vi faktiskt simulerar, är det därför viktigt att vi även ändrar på objektet vi mäter resultatet på. Simulerar vi linje E, måste vi därför givetvis mäta resultatet för linje E och inte för linje B. Vill vi simulera mätstationerna måste vi också mäta mätstationerna, och inte de vanliga stationerna.

Vill vi simulera och mäta samtliga linjer är det också viktigt att vi beslutar oss för hur viktig respektive linje är för det totala resultatet. Eftersom linje C är dubbelt så lång som linje H, bör också dess regularitet väga dubbelt så tungt när vi räknar fram den totala regulariteten för alla linjer tillsammans. För att få ett rättvist resultat bestämde vi oss för att antalet stationer på linjen var ett riktigt mått på hur stor vikt den skulle ha för det totala resultatet.

Då vi också delar upp varje linje i tre delar, huvudmätstationer, mätstationer, och vanliga stationer, måste vi även här vikta resultat. Hade mätningen skett på alla stationerna, hade det varit enkelt. Då hade alla fått lika stor vikt, men då vi nu vill mäta på några enstaka av linjens stationer måste de också få en inbördes viktning. Vi bestämde oss för att återigen bedöma vikten med anseende på antalet underordnade stationer som ligger före respektive mätstation eller huvudmätstation. Bild 7 visar indelningen för linje E södergående riktning för huvudmätstationerna, Hillerød, Svanemøllen, Dybbølsbro, och Køge. Hillerød är dock ointressant i denna riktning. Viktningen blir då: Svanemøllen 7/24, Dybbølsbro 6/24, Køge 11/24.

5 Modellbeskrivning

Modellen vi använt för simulering är ett resultat av ett annat examensarbete av Patrik Bjärkeson och Björn Johnsson. Vi vill i detta kapitel kortfattat beskriva modellen och även redogöra för de förändringar vi infört.

5.1 Beskrivning av S-tågnätet i verkligheten

S-tågnätet täcker med sina 85 stationer och sju linjer stora delar av Köpenhamn och dess förorter. Tågen går från Køge i söder till Hillerød i norr, och hjälper dagligen cirka en kvarts miljon människor att förflytta sig. S-tågnätet är separerat från övrig tågtrafik och har en egen strömstandard och ett eget signalsystem. Varje dag mellan kl 05.00 och kl 01.00 är tågnätet i drift med högst belastning mellan 06.00 och 10.00.

De flesta linjerna har tåg som avgår var tionde minut. Linje Bx och Linje H avgår dock endast var tjugonde minut. Alla linjer stannar inte på alla stationer de passerar. Vilka de stannar vid framgår av Bild 9. Linje F använder ett separat spår som inte påverkar övriga linjer. När tågen kommit fram till slutstationen vänder de och påbörjar att åka tillbaka på linjen i motsatt riktning.

Av säkerhetsskäl ska tågen vara separerade med en minut vid ankomst till stationen. Vid några stationer, som till exempel Hellerup, gör man även en prioritering mellan tågen vid ankomst.

Det centrala avsnittet, som består av alla stationer mellan Svanemöllan och Dybbølsbro har endast ett spår i vardera riktningen. Här uppstår därför en del komplikationer. Den tidtabellerade restiden mellan två stationer i det centrala avsnittet måste dessutom vara densamma för samtliga linjer. Detta innebär för vår del att bufferttiden i detta område måste vara samma.

Vid Köpenhamn H och Svanemöllan finns tillgång till två spår trots att dessa ligger i centrala avsnittet.

Om ett tåg skulle vara mycket försenat har DSB:s trafikledning möjlighet att avlysa tåget för att minska risken för följdförseningar.

Bild 9: Schematisk karta över S-tågnätet

Bufferttid används tillsammans med minsta körtid och minsta stopptid för att skapa en tidtabell som tidigare nämnts under definitioner i kapitel 1. DSB har som mål att ha en bufferttid som totalt inte överstiger 14 procent av den sammanlagda minsta körtiden för hela linjen. Givetvis hade en större mängd bufferttid varit gynnsam för att minska antalet regularitetsbrott, men det hade också lett till en längre restid för samma sträcka. Att nämnas bör att det inte är vår uppgift att ha några åsikter om den procentsatsen utan bara optimera efter given mängd bufferttid. I Tabell 6 visas hur många procent bufferttid varje linje har. Bufferttid efter halvminutsavrundning är det som DSB har satt sitt mål efter. Alla linjer uppnår inte kriteriet.

Linje	Bufferttid i procent av minsta körtid	Bufferttid efter halvminutsavrundning i procent
Linje ES	17,8%	13,2%
Linje EN	18,8%	15,7%
Linje BS	20,7%	15,2%
Linje BN	20,2%	15,5%
Linje AS	20,6%	16,3%
Linje AN	21,4%	16,7%
Linje BxS	16,2%	11,4%
Linje BxN	18,7%	14,1%
Linje HS	16,8%	13,1%
Linje HN	18,1%	16,0%
Linje CS	18,0%	14,1%
Linje CN	17,4%	14,4%

Tabell 6: Tabell över andelen bufferttid för de olika linjerna. Samt efter justering för halvminutsavrundning

5.2 Beskrivning av S-tågnätet i modellen

Nedan följer en grov beskrivning av modellen. För mer detaljer hänvisar vi till *”Manual för simuleringsmodellen som beskriver DSB:s S-tåg i Köpenhamn”.*

Modellen är gjord i Extend och efterliknar med tillräckligt önskvärd precision verkligheten. Den är testad och validerad och vi har förutsatt att den ger oss korrekta värden. Den fungerar i stort sett exakt som beskrivningen av S-tågnätet i 5.1, men med några modifikationer.

Alla linjer utom linje F är representerade då den inte påverkar systemet som helhet. Simuleringen sker under rusningstiden, alltså mellan 06.00 och 10.00, och inte under hela dagen. Modellen medger inte möjlighet att avlysa tåg för att minska risken för följdförseningar, och tar heller inte hänsyn till extraordinära händelser såsom olyckor och dylikt. Den använder endast stokastik vid på- och avstigningar medan körtiden alltid är densamma. Vid Köpenhamn och Svanemöllan har modellen endast ett spår i vardera riktningen, medan det i verkligheten är två spår här. Bild 10 nedan visar en översiktsbild av modellen i Extend. Södergående linjer startar till vänster och norrgående till höger, med samma intervall som i verkligheten. Tågen följer den förutbestämda tidtabellen och vid varje station som tågen stannar vid görs en kontroll huruvida de är i tid eller ej. Regularitetsbrott rapporteras och räknas samman. Samtliga linjer går samman i det centrala partiet.

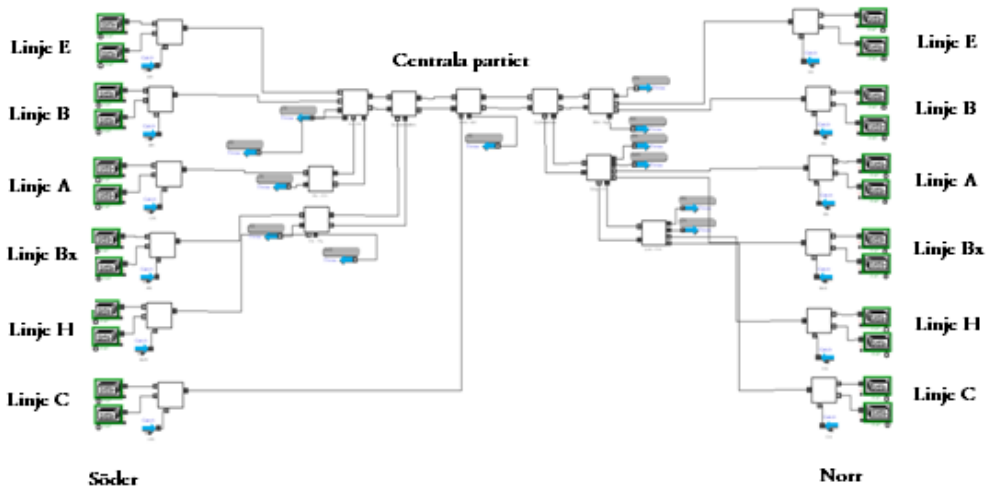


Bild 10: Översiktsbild över modellen i Extend

5.3 Förändringar av befintlig modell

Examensarbetets första uppgift var att förbättra infrastrukturen i modellen kring Dybbølsbro och Københavns HB. Idag har man i simuleringsmodellen gjort en förenkling kring Dybbølsbro (DBT) och Københavns HB (KH) och endast använt sig av ett spår. I verkligheten ser infrastrukturen något annorlunda ut.

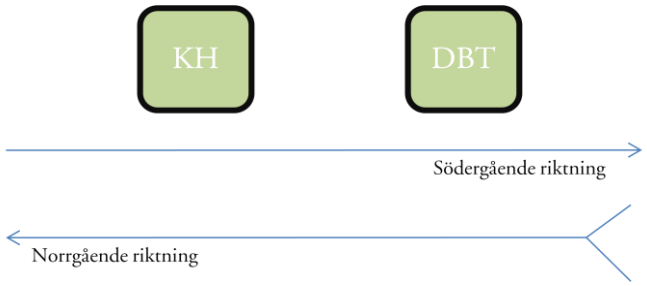


Bild 11: Infrastruktur vid Dybbølsbro, och Københavns HB i befintlig modell.

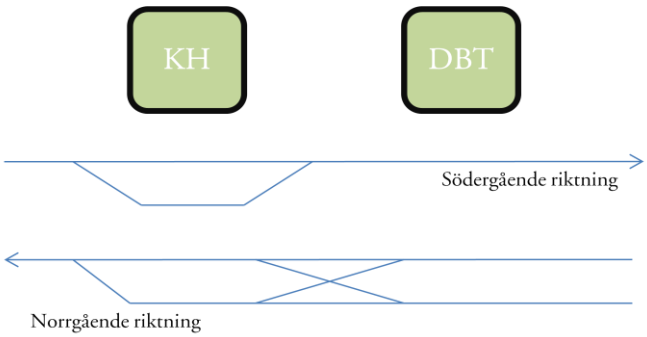


Bild 12: Infrastruktur vid Dybbølsbro, och Københavns HB i verkligheten.

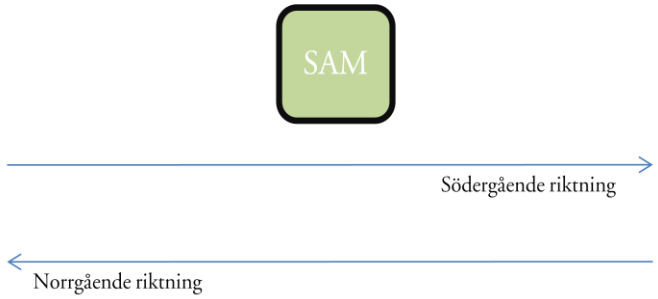


Bild 13: Infrastruktur vid Svanemøllen i befintlig modell.

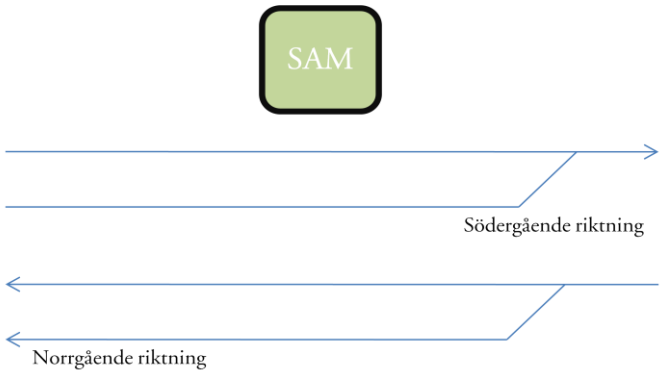


Bild 14: Infrastruktur vid Svanemøllen i verkligheten.

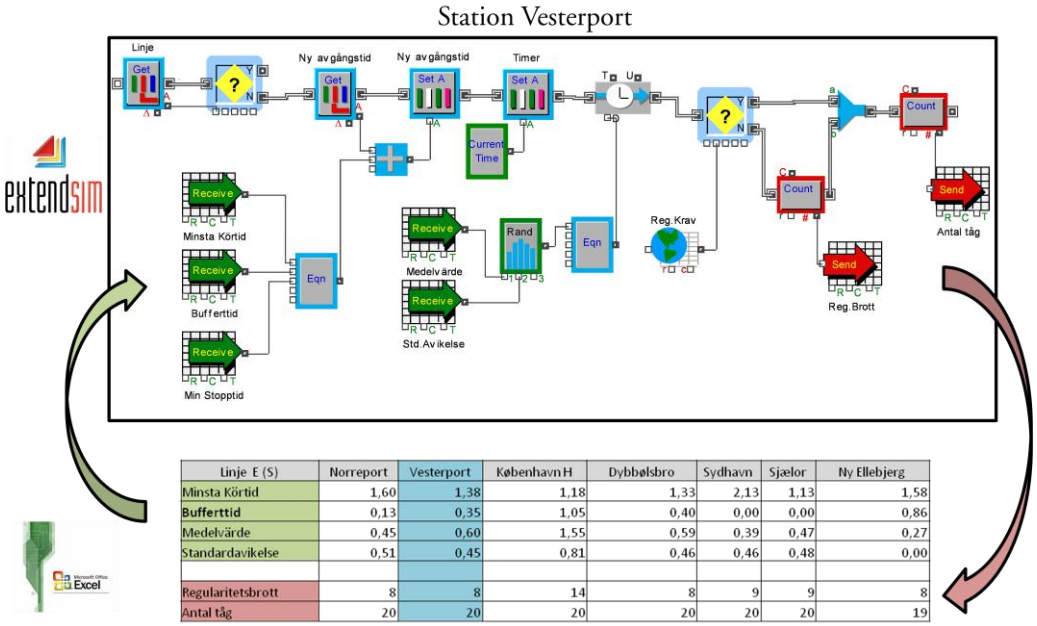
Problemen åtgärdades i Extend genom att bygga ut antalet spår där det behövdes. Den nya modellen testades och validerades. Då vi inte hade något att jämföra med när vi skulle validera, blev målet att få värdena på den nya modellen åtminstone bättre än på den gamla. På detta vis kunde vi åtminstone utesluta att vi gjort något grovt fel då vi ändrat i modellen. Resultatet blev en förbättring på ca 5 %, vilket får anses vara i linje med förväntningarna. Modellen var nu ännu mer lik verkligheten än vid starten.

6 Utförande

Den konceptuella idén beskrivs i kapitel 4. Här beskrivs mer i detalj hur det praktiskt utfördes.

6.1 Samspel mellan Extend och Excel

Utförandet har till stor del följt den konceptuella idén enligt Bild 3 i kapitel 4.2. Det första steget, modelleringen, klarades av i kapitel 5, och vi börjar därför beskriva samspelet, vid simulering, mellan Excel och modellen i Extend. I grund och botten är det ett stort antal experiment, eller simuleringar som ska utföras. Indata behandlas i modellen, och modellen svarar med utdata.



Linje E (S)	Norreport	Vesterport	København H	Dybbølsbro	Sydhavn	Sjælor	Ny Ellebjerg
Minsta Körtid	1,60	1,38	1,18	1,33	2,13	1,13	1,58
Bufferttid	0,13	0,35	1,05	0,40	0,00	0,00	0,86
Medelvärde	0,45	0,60	1,55	0,59	0,39	0,47	0,27
Standardavikelse	0,51	0,45	0,81	0,46	0,46	0,48	0,00
Regularitetsbrott	8	8	14	8	9	9	8
Antal tåg	20	20	20	20	20	20	19

Bild 15: Samspelet mellan Excel och modellen i Extend. Gröna parametrar är indata, och röda parametrar utdata. I exemplet är det stationen Vesterport som visas, och det är endast dess in- och utdata som förs in. Modellen är något modifierad på bilden.

All indata för samtliga linjer och stationer samlas i ett Excel dokument. För varje enskild station förs sedan dessa data över till respektive station i Extend. De indata vi använder oss av är följande:

- Minsta körtid

- Medelvärde för tiden för av- och påstigning
- Standardavvikelse för tiden för av- och påstigning
- Minsta stopptid (alltid 16 sekunder)
- **Bufferttid**

All indata utom bufferttiderna är konstanta. När indata för samtliga stationer förts över till Extend, utförs en simulering och resultatet rapporteras tillbaka till Excel. Resultat, eller utdata, som används är följande:

- Antal tåg
- Regularitetsbrott

Regulariteten är sedan ett direkt resultat av dessa utdata, då den är lika med andelen tåg som inte begår regularitetsbrott.

Bild 15 visualiserar hur detta fungerar för stationen i Vesterport. Förfarandet är i stort sett samma för alla stationer.

Efter att simuleringen är klar förs en ny kombination av bufferttid in för att simuleras i Extend, och så vidare, tills alla olika önskvärda kombinationer är simulerade.

6.2 Samspel mellan Excel och Visual Basic

Varje gång en ny kombination ska simuleras skulle det givetvis vara möjligt att föra in den manuellt i Excel, och sedan skriva ner resultatet i ett annat Excelblad. Det blir dock ett väldigt omfattande arbete om det ska utföras på 10 000 simuleringar, och därför automatiserade vi denna process. Processen visualiseras i Bild 16.

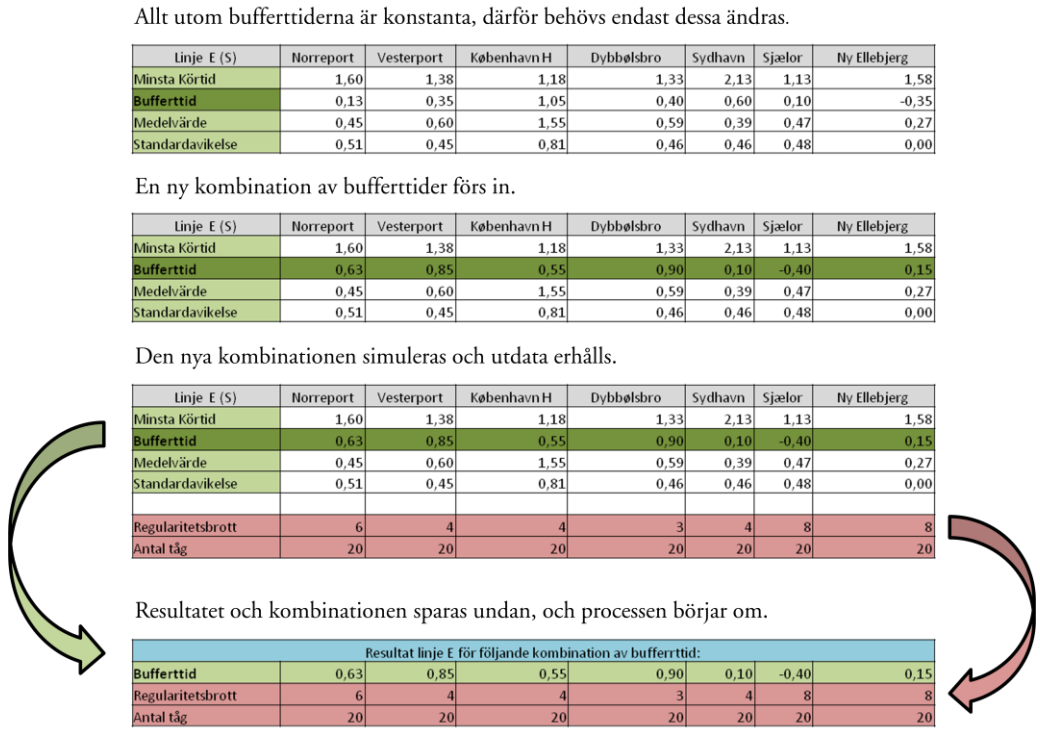


Bild 16: Beskriver utbytet av bufferttiderna i Excel samt tillvaratagandet av resultatet.

Vid automatiseringen tog vi hjälp av programmet Visual Basic for Applications, som är ett kraftfullt verktyg för att göra just sådana applikationer. I Visual Basic for Applications skrevs programkod, eller så kallade makron, där samtliga processer vi ville automatisera beskrevs och utfördes om och om igen. Då detta varit en stor del av examensarbetet presenterar vi i kapitel 6.2.2 samtliga använda makron, men för att förstå den delen behöver man först läsa nästa kapitel.

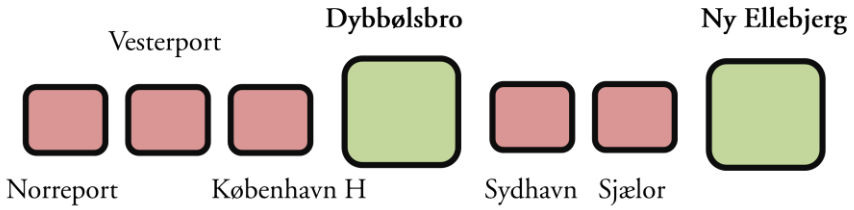
6.2.1 Bufferttider vid mätstationer och huvudmätstationer

Dagens tidtabell har bufferttid utspridd på samtliga stationer. För vår del har det dock endast varit intressant att testa och variera bufferttiden vid mätstationerna och huvudmätstationerna. Därför reduceras bufferttiderna på alla vanliga stationer till noll, och all tid adderas till respektive mätstation eller huvudmätstation. Bild 17 visar hur detta ser ut i Excel.

Alla stationer har en bufferttid om alla betraktas som vanliga stationer.

Linje E (S)	Norreport	Vesterport	København H	Dybbølsbro	Sydhavn	Sjælor	Ny Ellebjerg
Bufferttid	0,63	0,85	0,55	0,90	0,10	0,60	0,15

Vi inför ett antal mätstationer enligt nedan.



Nu adderas bufferttiden för de vanliga stationerna till mätstationerna.

Linje E (S)	Norreport	Vesterport	København H	Dybbølsbro	Sydhavn	Sjælor	Ny Ellebjerg
Bufferttid	0,00	0,00	0,00	2,93	0,00	0,00	0,85

Bild 17: Vid mätstationerna adderas samtliga bufferttider från föregående stationer.

Detta innebär att det kommer uppstå väldigt mycket regularitetsbrott vid de vanliga stationerna, som inte har någon bufferttid alls, men eftersom vi endast mäter resultatet på mätstationerna och huvudmätstationerna spelar detta ingen roll.

6.2.2 Förklaring till använda makron

Denna del kan uppfattas som något svår att förstå, då det rör sig om ganska komplicerade beskrivningar av programmerade makron. Den finns för att det ska vara möjligt att mer på djupet förstå hur programmeringen, och vårt praktiska arbete gått till. Varje makro presenteras i fet stil med sitt namn. Denna del kan också fungera som manual för de som vill använda sig av den utvecklade modellen i ett senare skede.

SimuleraOchPresentera

Precis som namnet antyder innebär makrot simulering av en viss kombination av bufferttider samt presentation av resultatet precis så som Bild 16 beskriver. Den raden med bufferttider som står skriven i Excel levereras alltså vidare till Extend för att där simuleras. Resultatet presenteras sedan av subprogrammen som makrot använder. Dessa

är: *StartaExtend, RegularitetHuvudmätstationer, RegularitetMätstationer, RegularitetAllaStationer.*

StartaExtend

StartaExtend är ett färdigskrivet makro som utvecklarna av Extend tagit fram för att kunna styra Extend via Excel. Det blir alltså själva kopplingen mellan de båda programmen. Detta makro anropas av *SimuleraOchPresentera.* Makrot kan starta Extend och den givna filen, men vi har märkt att det fungerar bättre om både Extend och filen med modellen redan är öppna.

RegularitetHuvudmätstationer

RegularitetHuvudmätstationer används av *SimuleraOchPresentera*. Det räknar ut regulariteten när vi bara har bufferttid på huvudmätstationerna. Det räknar helt enkelt ut regulariteten för varje huvudmätstation för varje linje. Sedan viktas den med så många vanliga stationer det finns innan. Till exempel, om linje E innehåller totalt 24 stationer och det finns 7 stationer innan det centrala-avsnittet kommer vikten för den huvudmästationens regularitet vara 7/24. Detta finns mer beskrivet i kapitel 4.6. Resultaten som detta makro räknar ut hamnar sedan på en separat flik i Excel där optimeringsprogrammen som *HuvudstationerBredd* och *SållningHuvudstationerDjup* kan hämta resultat ifrån.

RegularitetMätstationer

Detta makro används också av *SimuleraOchPresentera*. Det räknar ut regulariteten när vi endast har bufferttid på mätstationerna. Det räknar helt enkelt ut regulariteten för varje mätstation för varje linje. Sedan viktas den med så många vanliga stationer det finns innan. I detta makro behövs ingen sammanslagning av regulariteten för alla linjer utan det är bara varje linje som är intressant. Resultaten som detta makro räknar ut hamnar sedan på en separat flik i Excel där optimeringsprogrammen för varje linje, det vill säga makrot Optimera*MätstationerE-H,* kan hämta resultat ifrån.

RegularitetAllaStationer

Behöver vi någon gång mäta regulariteten där alla stationer räknas med använder vi oss av detta makro. Det anropas precis som de andra regularitetsmakrona med *SimuleraOchPresentera* och resultaten presenteras i en separat flik i Excel. Här mäts stationerna var för sig så ingen viktning behövs.

KöraBefintligt

Detta är ett mycket enkelt program som har till enda uppgift att köra en tidtabellskombination som finns inlagrad ett bestämt antal gånger. Makrot presenterar sedan önskvärt resultatet, det vill säga regulariteten, genom att använda sig av *RegularitetHuvudmätstationer, RegularitetMätstationer* eller *RegularitetAllaStationer.* Detta makro är användbart om man vill testa en kombination eller vill provköra en idé.

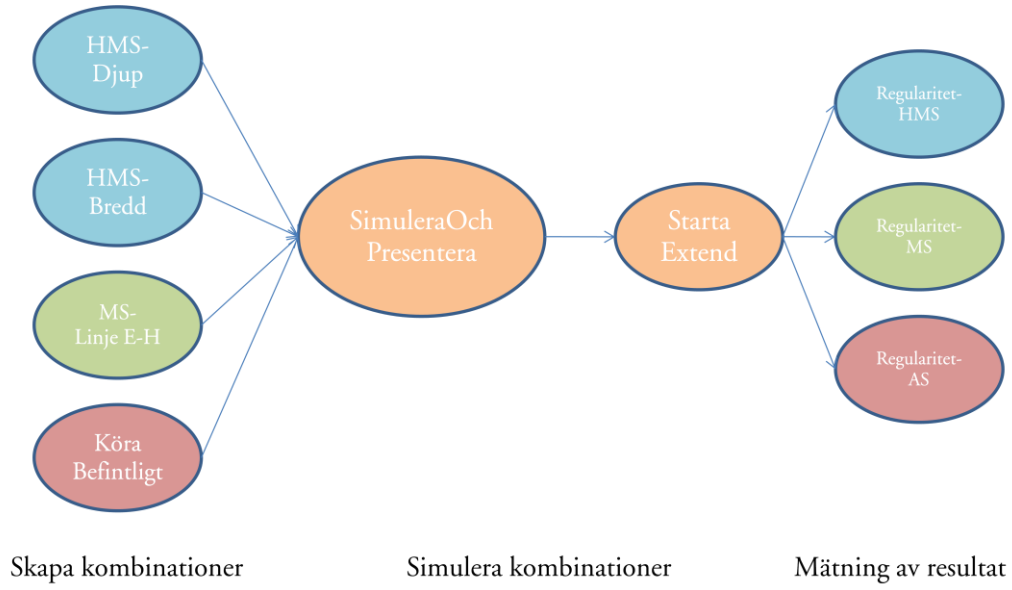


Bild 18: Översiktsbild över de viktigaste makrona som använts i examensarbetet.

HuvudMätStationerBredd

Detta Makro är till för att optimera fram en grundinställning för det centrala avsnittet. Den testar kombinationer av bufferttider i ett brett intervall för att kontrollera ungefär var den rätta kombinationen finns. Det fungerar så att den samlar alla huvudmätstationer för alla linjer och gör samma justering i bufferttid på alla. Det vill säga, minskas tiden med en minut innan det centrala avsnittet på LinjeE minskar även de andra fem linjernas bufferttid med lika mycket. Se 4.3.3 för mer information. Det är ju dock inte säkert att den optimala inställningen för linje E är samma som för linje B, men för att få fram en grundinställning är detta ett bra tillvägagångssätt. Detta makro skapar således en

kombination av bufferttidsförändringar från den befintliga tågtidtabellen. En sådan kombination kan vara:

-3	-2	5

Det betyder att bufferttiden mellan startstationen och stationen innan det centrala avsnittet minskats med tre halvminuter, jämfört med den befintliga tidtabellen. Bufferttiden på det centrala avsnittet har minskats med två halvminuter och på det sista avsnittet, har den ökat med fem halvminuter. Justeringarna som gjorts summerar alltid till noll vilket betyder att den totala bufferttiden för varje linje inte ändras.

Varje justering kan gå från minus fem till plus fem halvminuter. När väl kombinationerna är skapade kopieras i tur och ordning inställningen till varje linje och med de inställningarna körs sedan Extend med hjälp av *SimuleraOchPresentera*. Varje kombination körs 100 gånger, och ett medelvärde beräknas. Mätningen sker med hjälp av *RegularitetHuvudmätstationer.*

Av de 100 simuleringarna på varje linje tas sedan ett medelvärde ut, och då kan vi jämföra de olika inställningarna mot varandra. Den vinnande kombinationen kan till exempel ha blivit:

-2	3	-1

Då vet vi att dagens tidtabell borde justeras så att det blir 3 halvminuter mer i det centrala avsnittet, två halvminuter mindre innan och en halvminut mindre efter det centrala avsnittet.

HuvudMätStationerDjup

HuvudMätStationerBredd optimerade alla linjer samtidigt för att se om det fanns några generella fel i tidtabellen. Det användes för att hitta en grov inställning. Det behövdes nu ett program som fortfarande testade alla linjer samtidigt men kunde justera för variationer mellan de olika linjerna och som testade lite mer på djupet. Makrot fungerar så att det för varje linje skapar en mängd kombinationer där resultaten från *HuvudMätStationerBredd* användes som grund. Till exempel, om tiden före det centrala avsnittet för en linje i den

befintliga tidtabellen var sju halvminuter, och vi använder oss av resultatet från *HuvudMätStationerBredd*, är nu den tänkta ideala tiden fem halvminuter.

Spridningen i bufferttid för varje huvudstation blir den nya ideala tiden plus/minus ett lämpligt intervall. Detta intervall är beroende av om det finns andra kombinationer med nästan lika bra regularitet som den bästa kombinationen. I Tabell 7 ser vi möjliga kombinationer för linje E. Alla kombinationer summerar till samma tal, vilket är den tillgängliga bufferttiden i halvminuter som finns på linjen.

LinjeE Kombination	Innan det Centrala-Avsnitter	Centrala-Avsnittet	Efter det Centrala-Avsnittet
1	2	6	8
2	2	7	7
3	3	5	8
4	3	6	7
5	3	7	6
6	4	4	8
7	4	5	7
8	4	6	6
9	4	7	5
10	5	4	7
11	5	5	6
12	5	6	5
13	5	7	4

Tabell 7: Möjliga kombinationer för linje E, som summerar till 16 halvminuter.

När alla kombinationer skapats för varje linje behöver det skapas en komplett tidtabell för hela tågnätverket. Då gör vi så att vi tar en kombination från en linje och ser om den har samma tid på det centrala avsnittet som de andra fem linjerna. Uppfylls detta krav kan en kombination skapas med en tidtabell för alla linjers huvudmätstationer.

Linje	Innan det Centrala-Avsnitter	Centrala-Avsnittet	Efter det Centrala-Avsnittet
Linje E	4	5	7
Linje B	2	5	5
Linje A	4	5	6
Linje Bx	1	5	4
Linje H	0	5	5
Linje C	0	5	10

Tabell 8: Matris som beskriver en inställning för samtliga linjer, där allt utom det centrala avsnittet tillåts variera inbördes.

Således kommer det bli en stor mängd tidtabellskombinationer i och med att alla individuella linjers kombinationer måste simuleras mot alla de andra. När väl en tidtabells kombination är skapad, kopieras värdena för respektive linje tillsammans med avrundningsbufferten, som sedan simuleras i Extend. Resultatet presenteras på samma sätt som för *HuvudMätStationerBredd,* det vill säga det viktade medelvärdet av regularitet för Huvudmätstationerna.

SållningHuvudMätStationerDjup

Antalet kombinationer på *HuvudMätStationerDjup* blev ganska många, så vi var tvungna att utveckla en modell som sållar bort de kombinationer som var så dåliga att vi redan i ett tidigt stadium kunde förkasta dem och lägga resurser på att simulera de kombinationer som var bra istället. Om en kombination av bufferttider vid första simuleringen visat sig vara markant sämre än den bästa, är det lika bra att strunta i de resterande 99 simuleringarna för just den kombinationen. Den matematiska modellen för detta finns presenterad i kapitel 6.3. Där beskrivs hur selekteringen går till. Rent praktiskt fungerar den så att vi kör *HuvudMätStationerDjup* först så att vi får minst två resultat på varje kombination. *SållningHuvudMätStationerDjup* kommer sedan successivt att ta bort de kombinationer som inte anses tillräckligt bra att köra vidare. Den första tanken med detta makro var att det skulle köras tills det bara fanns en vinnande kombination kvar, men det blir praktiskt väldigt svårt att styra så därför satte vi ett max antal simuleringar till 100 stycken.

OptimeraMätStationerE-H

När det centrala avsnittets bufferttider var bestämda, kunde vi gå vidare och börja simulera varje linje för sig. Optimera*MätStationerE* är till för att optimera bufferttiden för mätstationerna på Linje E, men det finns givetvis ett program för varje linje ända till linje H. Det fungerar så, att vi kan variera bufferttiden på varje mätstation med plus/minus ett visst antal halvminuter från dagens tidtabell, med bivillkoret att de summerar till den totala tillgängliga bufferttiden för hela linjen. Det har också tillkommit bivillkor i och med att alla mätstationers bufferttider för, under och efter det centrala avsnittet måste vara lika med det värde som bestämdes i *HuvudMätStationerDjup.* I Tabell 9 ser vi hur det kan se ut.

Avsnitt	**Innan det centrala avsnittet**					**Centrala-Avsnittet**	**Efter det centrala-avsnittet**	
Bivilkor från HMSDjup	4					5	7	
Mästationer	**Birker.**	**Holte**	**Lyngby**	**Heller.**	**Svanem.**	**Dybbølsbro**	**Ellebjerg**	**Køge**
Bufferttids-kombination	1	-1	1	2	1	5	4	3
Bufferttids-kombination	2	-1	2	2	-1	5	5	2

Tabell 9: Visar två kombinationer av bufferttider för linje E, med bivillkor i form av resultatet från HuvudMätStationerDjup. Summan av bufferttiderna summeras till 16.

När alla möjliga kombinationer är skapade kopieras var och en till sin linje och körs i Extend i tur och ordning. Vi kör 50 simuleringar på varje kombination för att få bort de slumpmässiga variationerna. De utdata vi får är ett viktat värde av regulariteten för just den linje vi optimerar.

NollaAllt

Vi hade problem med att det ibland fanns kvar bufferttid från tidigare körningar när vi ville testa nya värden. För att säkerställa att det inte hade glömts bort att radera några bufferttider konstruerade vi detta makro som raderar alla bufferttider för alla linjer så att endast de tider som vi vill ha där, för den aktuella optimeringen används. Alla optimeringsmakron börjar därför med *NollaAllt*.

KopieraMätstationerna

När alla stationer var optimerade vid mätstationerna behövde de köras tillsammans. De skulle kunnat gå att flytta alla de ”vinnande” kombinationerna manuellt, men för att undvika fel så gjorde vi så att alla de bästa bufferttidskombinationerna lades i en flik i Excel, som sedan kopierades med detta makro till rätt plats och linje. Det totala resultatet fås sedan genom att köra makrot, *KöraBefintligt.*

6.3 Sållningsmodell

För att minimera antalet nödvändiga simuleringar, när vi skulle finna den bästa inställningen för bufferttiderna, utvecklade vi en sållningsmodell. Makrot som utför denna åtgärd heter: *SållningHuvudMätStationerDjup.* Den sållar bort och kör inte de inställningar, som är så dåliga att de i ett tidigt stadium kan förkastas. På så sätt spar vi simuleringstid och kan köra de inställningar som ger bra resultat fler gånger. Modellen fungerar så här:

Antag att $X_{i,j}$ är regulariteten hos den j:te simuleringen av bufferttidsinställningen i. Vi börjar då med att skapa följande matris.

i/j	1	2
1	$X_{1,1}$	$X_{1,2}$
2	$X_{2,1}$	$X_{2,2}$
3	$X_{3,1}$	$X_{3,2}$
…	…	…
n	$X_{n,1}$	$X_{n,2}$

Här ser vi att vi har kört alla olika bufferttidsinställningar två gånger. Medelvärde och standardavvikelse beräknas sedan för varje inställning enligt nedan, till exempel så är $\bar{X}_{1,1-2}$ medelvärdet på regulariteten för bufferttidsinställningen 1 och av körningarna 1 och 2. Samma notation gäller även för standardavvikelsen.

i/j	1	2	$Medelvärde_2$	Standardavvikelse $_2$
1	$X_{1,1}$	$X_{1,2}$	$\bar{X}_{1,1-2}$	$\sigma_{1,1-2}$
2	$X_{2,1}$	$X_{2,2}$	$\bar{X}_{2,1-2}$	$\sigma_{2,1-2}$
3	$X_{3,1}$	$X_{3,2}$	$\bar{X}_{2,1-2}$	$\sigma_{3,1-2}$
…	…	…	…	…
n	$X_{n,1}$	$X_{n,2}$	$\bar{X}_{n,1-2}$	$\sigma_{n,1-2}$

När två körningar av varje inställning är genomförda är då frågan om det är nödvändigt att köra X_i en tredje gång. Vi jämför då $\bar{X}_{,1-2}$ mot $\max(\bar{X}_{1,1-2}, \bar{X}_{2,1-2}, \dots, \bar{X}_{n,1-2})$, och kör sedan ett hypotestest på detta där vi som standardavvikelse använder $\max(\sigma_{1,2}, \sigma_{2,2}, \dots, \sigma_{n,2})$, detta för att vara på den säkra sidan och inte på något sätt underskatta standardavvikelsen. Hypotestestet ser då ut enligt följande.

$$\lambda_i = \frac{\max(\bar{X}_{1,1-2}, \bar{X}_{2,1-2}, \dots, \bar{X}_{n,1-2}) - \bar{X}_{i,1-2}}{\frac{\sqrt{2} \cdot \max(\sigma_{1,1-2}, \sigma_{2,1-2}, \dots, \sigma_{n,1-2})}{\sqrt{n}}}$$

Om λ_i är större än avståndet från det tänkta maxmedelvärdet för givet α, vårt fall 1,96 då vi använder en signifikans på 95 procent, förkastas kombinationen, det vill säga kombinationen ger så dålig regularitet redan efter två körningar att vidare körningar inte anses nödvändigt. Är $\lambda_i$mindre kan vi inte i detta stadium säga med givet α att $\bar{X}_{,1-2}$ är sämre än $\max(\bar{X}_{1,1-2}, \bar{X}_{2,1-2}, \dots, \bar{X}_{n,1-2})$ och kombinationen X_i måste köras en tredje gång. Att det är $\sqrt{2}$ i nämnaren beror på att det är två normalfördelningar som jämförs mot varandra. Detta beror på att vi vet att skillnaden mellan två normalfördelningar, $X \in N(\mu_x, \sigma_x)$ och $Y \in N(\mu_y, \sigma_y)$ kan skrivas som:

$X - Y \in N\left(\mu_x - \mu_y, \sqrt{\sigma_x^2 + \sigma_y^2}\right)$ och i och med att vi använder $\sigma_{\max}$ som standardavvikelse för både dem vi jämför emot kan vi sätta $\sigma_{max} = \sigma_x = \sigma_y$ och då blir $\sqrt{\sigma_x^2 + \sigma_y^2} = \sqrt{2} \cdot \sigma_{max}$.

När alla bufferttidsinställningar som klarade hypotestesten körts en tredje gång uppdateras medelvärde och standardavvikelse, således uppdateras även maxvärdet för medelvärde och standardavvikelse . Ett nytt hypotestest genomförs och en fjärde körning görs på de inställningar som ligger inom konfidensintervallet. Så här fortsätter testet tills det bara är en vinnande kombination av bufferttider kvar. Tyvärr är det så för vissa körningar att olika inställningar ger väldigt lika resultat och det blir då inte bara en kvar. Då har vi varit tvungna att sätta ett max antal körningar på till exempel 100 stycken.

Det smarta med detta sätt är, att $\max(\sigma_{1,1-j}, \sigma_{2,1-j}, \dots, \sigma_{,1-j})$ konvergerar mot den sanna standardavvikelsen och hela tiden blir mindre, samt att $\sqrt{n}$ blir större med ökande

antal körningar. På så sätt krymper intervallet för hypotestestet med varje körningsomgång och fler inställningar kan sållas.

6.4 Minimering av simuleringstiden

En tanke vi hade redan från början var, att om vi kunde minimera exekveringstiden för modellen, hade det gjort mycket för det totala resultatet. Dock insåg vi snabbt att det var simuleringstiden för varje simulering som var det intressanta att reducera. Detta lyckades vi över förväntan med. Simuleringstiden för en simulering var cirka 1 minut och 46 sekunder då den var som mest under arbetets gång. Efter några kraftfulla åtgärder lyckades vi dra ner den till cirka 12 sekunder. Det är en minskning med nästan 90 procent, och det möjliggör givetvis simulering av många fler kombinationer. Kunde vi tidigare utföra 10 000 simuleringar under en viss tid, kunde vi nu helt plötsligt utföra nästan 100 000. Åtgärderna som möjliggjorde detta var följande:

- Separering av norrgående och sydgående linjer i modellen. Då vi ändå optimerade dem var för sig var det helt onödigt att låta norr köra samtidigt som söder. Detta halverade nästan tiden.
- Uppgradering av programvaran för Extend från 6.0 till 7.0
- Borttagande av utdata som skickades från Extend till Excel som ändå inte skulle mätas, så som ackumulerad försening.
- Borttagande av indata som skickades till Extend från Excel varje gång och som ändå var konstant, till exempel minsta stopptid, som aldrig ändrades. Även bufferttiderna vid centrala avsnittet räckte det att skicka en gång, då de ju faktiskt var samma för samtliga linjer.

Endast separeringen av norrgående och sydgående linjer utgör en förändring av själva modellen. Den ändringen har vi därför validerat i kapitel 1.5.

6.5 Simulering i praktiken

Verktygen för att utföra uppgiften var nu på plats, och det var dags att börja simulera. Då det tagit ganska lång tid att utveckla alla makron, och dra ner på simuleringstiden för varje enskild simulering, fanns det inte så mycket tid kvar, men eftersom varje simulering nu gick nästan 10 gånger så snabbt, hann vi ändå med fler simuleringar än vi först räknat med. Cirka 160 000 simuleringar utfördes, och med de resurser vi haft tycker vi det är ett godkänt resultat. Beräkningsservrarna på Lunds tekniska högskola har inte varit några hypermoderna datorer, och de låste sig mitt under pågående simulering. Därför blev vi tvungna att övervaka processen lite mer än vi trodde från början.

7 Resultat

I detta kapitel presenteras de resultat vi kommit fram till under examensarbetet.

7.1 Kommentarer till syftet

I syftet beskrevs en önskan om att finna en tidtabell som gav mindre försening än dagens. Istället för försening har vi använt oss av regularitet, som är definierat i avsnitt 1.4. En simuleringsheuristik som möjliggjorde en minskning av antalet teoretiska tidtabeller, samt en reducering av modellens körtid per simulering, var också önskvärda vid arbetets begynnelse. Alla tre anser vi oss ha lyckats med. Nedan följer resultatet.

7.2 Dagens värden

Regulariteten för mätstationerna för den tidtabell som används idag, presenteras nedan i Tabell 10. Södergående riktning.

Linje	Regularitet
Linje E(s)	90,73% - 91,47%
Linje B(s)	82,20% - 83,15%
Linje A(s)	91,22% - 91,89%
Linje Bx(s)	85,86% - 86,88%
Linje H(s)	75,86% - 77,55%
Linje C(s)	77,88% - 78,99%

Tabell 10: Resultatet för regulariteten på mätstationerna för dagens tidtabell. Presenterat med konfidensintervall på 95 procent. Södergående riktning.

Regulariteten för mätstationerna för den tidtabell som används idag presenteras nedan i Tabell 11. Norrgående riktning.

Linje	Regularitet
Linje E(n)	91,58% - 93,08%
Linje B(n)	83,62% - 85,72%
Linje A(n)	88,10% - 89,75%
Linje Bx(n)	71,64% - 75,51%
Linje H(n)	83,63% - 85,78%
Linje C(n)	77,16% - 79,68%

Tabell 11: Resultatet för regulariteten på mätstationerna för dagens tidtabell. Presenterat med konfidensintervall på 95 procent. Norrgående riktning.

Värdena har tagits fram genom simulering av dagens tidtabell i modellen. Det är alltså inte regulariteten i verkligheten, utan regulariteten för verklighetens indata i modellen. Detta för att möjliggöra en jämförelse med vårt resultat.

7.3 Vårt resultat

Efter simulering av en stor mängd tidtabeller kunde en väljas ut som den bästa. Resultatet presenteras nedan, med en jämförelse med dagens värden.

7.3.1 Södergående linjer

Regulariteten för den kombination som gav bäst resultat för södergående linjer presenteras nedan i Tabell 12.

Linje	Regularitet
Linje E(s)	91,30% - 92,92%
Linje B(s)	83,39% - 85,71%
Linje A(s)	92,62% - 94,06%
Linje Bx(s)	86,52% - 88,55%
Linje H(s)	77,33% - 80,35%
Linje C(s)	81,36% - 83,95%

Tabell 12: Det slutgiltiga resultatet för regularitet på mätstationerna. Presenterat med konfidensintervall på 95 procent.

Skillnaden mellan vårt optimerade resultat och den tidtabell som idag finns på mätstationerna presenteras nedan i Tabell 13.

Linje	Förändring	Konfidens-Intervall
Linje E(s)	1,01%	-0,16% - 2,17%
Linje B(s)	1,87%	0,29% - 3,45%
Linje A(s)	1,79%	0,74% - 2,83%
Linje Bx(s)	1,17%	-0,37% - 2,70%
Linje H(s)	2,14%	-0,31% - 4,58%
Linje C(s)	4,22%	2,41% - 6,03%

Tabell 13: Visar resultatförändring av regulariteten för de södergående linjerna. Presenterat både med medelförbättring och med ett konfidensintervall på 95 procent

7.3.2 Norrgående linjer

Regulariteten för den kombination som gav bäst resultat för norrgående linjer presenteras nedan Tabell 14.

Linje	Regularitet
Linje E(n)	91,06% - 92,67%
Linje B(n)	85,67% - 87,43%
Linje A(n)	90,39% - 92,09%
Linje Bx(n)	77,5% - 80,72%
Linje H(n)	79,94% - 81,93%
Linje C(n)	78,44% - 79,94%

Tabell 14: Det slutgiltiga resultatet för regularitet på mätstationerna i norrgående riktning. Presenterat med konfidensintervall på 95 procent.

Skillnaden mellan vårt optimerade resultat och den tidtabell som idag finns på mätstationerna presenteras nedan i Tabell 15.Tabell 2

Linje	Förändring	Konfidens-Intervall
Linje E(n)	-0,46%	-1,56% - 0,63%
Linje B(n)	1,88%	0,51% - 3,24%
Linje A(n)	2,32%	1,14% - 3,51%
Linje Bx(n)	5,53%	3,02% - 8,05%
Linje H(n)	-3,77%	-5,24% - -2,31%
Linje C(n)	0,77%	-0,69% - 2,23%

Tabell 15: Visar resultatförändring i procent av regulariteten för de södergående linjerna. Presenterat både med medelförbättring och med ett konfidensintervall på 95 procent.

7.4 Antalet simuleringar

Ett av de stora målen var att hitta en bra simuleringsheuristik och en stor del av detta var att drastiskt minska antalet simuleringar. Det antal simuleringar vi fick till slut för de södergående och norrgående linjerna kan ses nedan i Tabell 16 och Tabell 17. Det bör nämnas att antalet körningar per kombination för djupsökningen bara är ett snitt, på grund av att vi använde oss utav en sållningsalgoritm.

	Först		Linjerna					
	Breddsökning	Djupsökning	Linje E(s)	Linje B(s)	Linje A(s)	Linje Bx(s)	Linje H(s)	Linje C(s)
Kombinationer	91	2323	135	48	102	60	46	48
Körningar/Kombination	40	25	50	50	50	50	50	50
Summa simuleringar	3640	58476	6750	2400	5100	3000	2300	2400
Summa totalt	**84066**							

Tabell 16: De antal simuleringar som krävdes för att komma fram till resultatet för de södergående linjerna.

	Först		Linjerna					
	Breddsökning	Djupsökning	Linje E(n)	Linje B(n)	Linje A(n)	Linje Bx(n)	Linje H(n)	Linje C(n)
Kombinationer	91	2324	490	64	266	325	141	141
Körningar/Kombination	40	20	20	50	50	50	50	50
Summa simuleringar	3640	47069	9800	3200	13300	16250	7050	7050
Summa totalt	**107359**							

Tabell 17: De antal simuleringar som krävdes för att komma fram till resultatet för de norrgående linjerna.

Antalet simuleringar totalt blev därför cirka 200 000. Det kan jämföras med de 10^{252} kombinationer som skulle simuleras 100 gånger om samtliga kombinationer skulle testats. En klar framgång alltså. Reduceringen gör ju givetvis att resultatet inte med säkerhet kan sägas vara optimalt, men eftersom det är bättre än dagens tidtabell får optimeringsheuristiken anses lyckad.

7.5 Modellens exekveringstid

I syftet uttryckte vi en önskan om att dra ner modellens exekveringstid, men under arbetets gång fann vi att det egentligen var den totala tiden för att utföra en simulering som var intressant att titta på. Denna har vi minskat med cirka 90 %, vilket får ses som en rejäl förbättring. Om detta står det mer i kapitel 6.4.

8 Slutsats och diskussion

Kommentarer till resultatet, samt förslag till framtida förbättringar.

8.1 Kommentarer till resultatet

Syftet med detta examensarbete var att utveckla en simuleringsheuristik som gjorde det möjligt att använda simulering som verktyg för att skapa bättre tidtabeller. Det har alltså inte bara varit att nå målsättningen med att hitta en bättre regularitet än dagens tidtabell medger, utan även vägen till det målet. Det enda sättet vi kan utvärdera om simuleringsheuristiken är bra är att se om regulariteten förbättrades. För de södergående linjerna kan vi se förbättring på alla linjer, vilket tyder på att heuristiken uppfyller sitt mål och tar fram en bättre tidtabell. Även i norrgående riktning kan vi se en förbättring, även om några linjer blivit sämre.

Resultaten borde kunna förbättras genom att testa fler kombinationer med större spann men detta kräver mer datorkraft, vilket har varit ett problem för oss. För vår uppdragsgivare borde det dock inte vara något problem att köpa in fler datorer, om det kan minska belastningen för de som jobbar med tidtabellsframtagningen.

Det bör poängteras att resultaten är baserade på mätstationerna, och inte på hela tidtabellen. Ett steg kvarstår alltså innan en färdig tidtabell kan erhållas. Detta steg är i stort sett redan gjort, då det var exakt vad föregående examensarbetare gjorde. Det är givetvis inte helt säkert att den totala tidtabellen kommer att bli bättre bara för att mätstationerna visade sig vara det, men det är mycket troligt.

Sammantaget tycker vi att resultatet var över förväntan, och hoppas det kan komma DSB till nytta.

8.2 Vad vill vi se i framtiden

Under ett examensarbete uppkommer alltid nya idéer och frågeställningar som det inte finns utrymme för att utreda under examensarbetes gång. Att de inte har utretts har ofta berott på att det helt enkelt inte har fått plats inom tidsramen, men ibland har det också saknats kunskap och kompetens för att förverkliga idén. Därför presenterar vi nedan lite idéer som kom upp under examensarbetes gång för att visa på de möjligheter det finns för DSB och Lunds Tekniska Högskola att följa upp detta projekt.

8.2.1 Parallellisering

Under vårt examensarbete har vi arbetat med *en* beräkningsserver. Detta har gjort att simuleringen tagit mycket tid i anspråk. Det som tar mycket tid med simuleringarna är att varje kombination av bufferttider måste simuleras flera gånger för att minimera slumpmässiga variationer och få ett bra medelvärde. En tanke för framtida arbete är att utröna de praktiska möjligheterna att köra simuleringarna på flera beräkningsservrar samtidigt, så kallad parallellisering. Rent teoretisk är vårt problem mycket enkelt att parallellisera, för ingen beräkning är beroende av någon annan beräkning i samma steg . Detta innebär att om vi skulle dubblera antalet beräkningsenheter skulle tiden för simuleringen halveras.

8.2.2 Iterering

Vi såg att efter det att vi optimerat varje linje för sig, tappar de i regularitet när alla linjer sedan ska köras tillsammans. En tanke hade då varit att först optimera linjernas mätstationer var för sig, precis som vi gör idag och sen köra en ny optimering med det nya resultatet som indata, med andra ord att utföra en iterering på resultatet. I princip kan detta göras hur många gånger som helst för att få ett så bra värde som möjligt. Detta är den enklaste formen av iterering av vår modell.

Idag använder vi oss som sagt av dagens tidtabell som grunddata, men till exempel om linjerna skulle läggas om eller det skulle behövas ett större avbrott för en reparation finns ingen tidtabell att utgå ifrån. Då är en tanke att använda hela vår heuristik ett antal gånger och iterera fram en bra lösning. Idag finns det begränsningar för hur mycket en lösning kan skilja sig från den befintliga tidtabellen men itereras den tillräckligt många gånger är distansen i princip hur stor som helst. Har inte DSB en tågtabell skulle de bara kunna skapa några godtyckliga bufferttider där bufferttiden är förslagsvis jämnt utspridd över

linjen. Sedan är det bara att iterera fram en lösning av bufferttid, antingen så länge det finns tid eller till en önskad regularitet uppnås. Vi anser att vår heuristik är konstruerad så att det inte finns någon risk för att gå en i ”återvändsgränd” så det finns ingen risk att det skulle bli sämre regularitet när man itererar fram en lösning.

8.2.3 Industriell anpassning

Idag när DSB lägger tidtabeller för S-tågen har vi förstått att detta är ett stort projekt som tar mycket tid och resurser i anspråk. Vi hävdar att DSB skulle kunna använda sig av både parallellisering och iterering och på så sätt effektivisera tidtabellsläggningen. Ett förslag till framtida examensarbete hade varit att undersöka möjligheterna för DSB att investera i, eller hyra in, beräkningskapacitet för att helt modernisera tidtabellsläggningen. För att ge ett exempel, har vi under detta examensarbete bara använt oss av *en* beräkningsserver, och vi uppskattar den totala tiden för simuleringarna till cirka 50 dygn. Skulle DSB investera i 100 stycken beräkningsserverar, för uppskattningsvis en halv miljon svenska kronor, skulle de således kunna nå samma resultat som vi på ett halvt dygn, och på två veckor skulle de kunna iterera lösningen 28 gånger. Detta tror vi kan vara ett bra och kostnadseffektivt komplement till hur tidtabellsläggningen fungerar idag.

8.2.4 Area-Minimering

En tanke vi fick ganska tidigt är att vårt mål med optimeringen att minska antalet regularitetsbrott. Ett regularitetsbrott är som bekant två och en halv minuts försening. Detta kan i praktiken innebära att vi har optimerat fram en lösning som i och för sig minskar antalet regularitetsbrott men de kan också ha ökat antalet tåg som inte går på utsatt tid. Och för DSB:s image är det kanske bra om man minskar förseningen mer generellt än bara försöker minimera regularitetsbrotten. Därför kom idén om *Area-Minimering* upp. Den innebär minimering av den ackumulerade förseningen, eller med andra ord den area som uppstår under förseningsgrafen i Bild 19. Detta hade på ett, enligt oss, riktigare sätt givit en bättre fördelning av bufferttider.

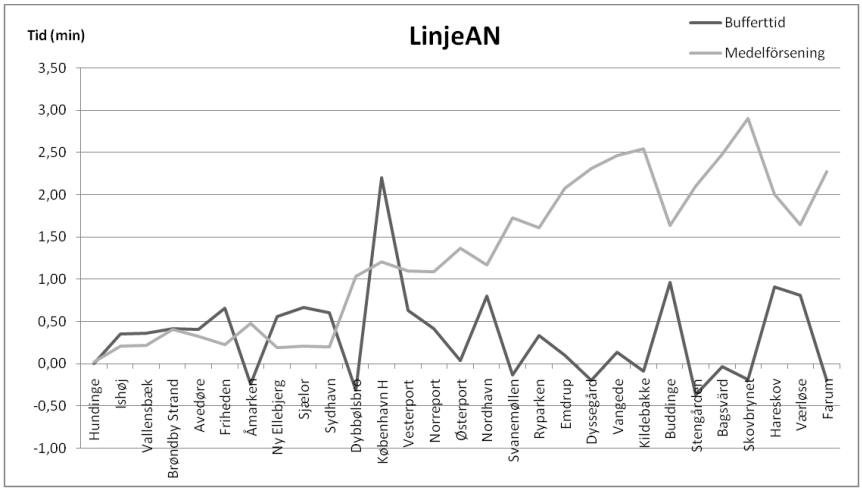


Bild 19: Den ljusa linjen visar den ackumulerade medelförseningen för varje station.

8.2.5 "Black-Box teorin"

En annan idé som kom upp var ett annorlunda sätt att se på hur optimering av bufferttiderna skulle gå till. Observera att förseningen är en funktion av tidtabellen och således fördelningen av bufferttiderna. Denna funktion känner vi inte till utan, det är en såkallad *"Black-Box"* som på ett okänt sätt förändrar insignalen, tidtabellen, till utsignalen, regulariteten. Ett sätt att försöka få fram denna funktion är att anta att om vi lägger mer och mer bufferttid på en station kommer regulariteten för hela systemet öka. Ökningen kommer antagligen vara avtagande det vill säga ha ett konkavt utseende. Vi skulle då kunna få fram varje stations "funktion" genom att ha alla stationers bufferttid till noll och bara öka den valda stationens bufferttid och simulera dess inverkan på den totala regulariteten. Sedan hade vi haft 85 stycken funktioner som tillsammans med givna bivillkor bildar den tänkta funktionen mellan tidtabell och regularitet. Den maximala regulariteten kan sedan beräknas med linjärprogrammering. Detta sätt kräver i och för sig också en mängd simuleringar men vi tror det kan vara ett effektivt sätt att komma nära en optimal lösning.

9 Referenser

I detta kapitel kan läsaren se en lista över de referenser vi använt under detta arbete.

9.1 Litteratur

Projektplan för BuffertSIMS fas 2 av Stefan Vidgren 2009-01-22

Att genomföra examensarbete Höst, Regnell, Runeson. 2006

Software process models, Sommerville Ian, ACM Computing Surveys, 1996

Sannolikhetsteori och statistikteori med tillämpningar, Gunnar Blom et al. 2005

Handbooks in OR & MS Vol. 4, S.C. Graves et al., Eds. 1993

Introduction to Operations Research. Hillier and Lieberman 2001

9.2 Elektroniska källor

http://www.dsb.dk/Om-DSB/Virksomheden/ 2009-04-22

http://www.extendsim.com/prods_overview.html 2009-05-05

http://office.microsoft.com/sv-se/excel/default.aspx 2009-05-05

http://msdn.microsoft.com/en-us/isv/bb190538.aspx 2009-05-05