HÖGSKOLAN SKÖVDE

Institutionen för kommunikation och information
Examensarbete i datalogi 20p
C-nivå
Vårterminen 2006

# Representation och algoritmer för optimering av postdistributionsnätverk

**Martin Andersson**

HS-IKI-EA-06-101

**Representation och algoritmer för optimering av postdistributionsnätverk**
Examensrapport inlämnad av Martin Andersson till Högskolan i Skövde, för Kandidatexamen (B.Sc.) vid Institutionen för kommunikation och information. Arbetet har handletts av Henrik Grimm.

**2006-06-07**

Härmed intygas att allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och att inget material är inkluderat som tidigare använts för erhållande av annan examen.

Signerat: _____

**Representation och algoritmer för optimering av postdistributionsnätverk**

**Martin Andersson**

# Sammanfattning

Det här arbetet undersöker om heuristik i en algoritm kan förbättra optimeringen av postdistributionsnätverk. Som optimeringsalgoritm används hill climbing och heuristiken appliceras på mutationsoperatorn . För att utvärdera mutationsoperatorn skapas en förenklad modell av ett postdistribueringsnätverk. Ett befintligt postdistribueringsnätverk används som utgångspunkt för den förenklade modellen för att snabbt kunna få en bra och realistisk modell. Resultaten av undersökningen indikerar på att den heuristiska mutationsoperatorn sänker tiden det tar att hitta en bra lösning jämfört med att använda en slumpmässig mutationsoperator.

**Nyckelord:** Optimering, Distributionsnätverk, Heuristik

# Innehållsförteckning

# 1 Introduktion

Ett distributionsnätverk består av olika system som sammanbinder producenter och konsumenter av varor. Dessa system kan t.ex. vara vattenledningsnätverk som förser en stad med vatten eller postnätverk som delar ut post i ett land. Det kan också vara en produktionskedja där ett företag har ett antal fabriker och lagerlokaler som transporterar delar mellan sig. En vara kan oftast transporteras på flera olika sätt genom de olika systemen. De olika transportsätten har fördelar och nackdelar, som exempelvis tid, kostnad och miljöpåverkan.

Distributionsnätverk finns överallt omkring oss. Det kan vara allt från den lokala blomsterhandlaren till internationella oljebolag. För att ett stort företag skall kunna vara konkurrenskraftigt måste det ha ett väldesignat distributionsnätverk och det måste även kunna utnyttja det på ett effektivt sätt. Det är inte bara ur ett kostnadsperspektiv som det kan vara aktuellt att effektivisera sitt distributionsnätverk, även tuffare miljöregler eller hårdare miljöpolicy inom ett företag kan vara orsaker.

## 1.1 Nivåer av planering

För att åstadkomma ett effektivt distributionsnätverk krävs planering. Planeringen kan ske på tre nivåer, strategisk (långsiktigt), taktisk och operativ (kortsiktigt) nivå (Crainic T.G, 2002)

Den strategiska planeringen sker långsiktigt och innefattar beslut så som hur det fysiska distributionsnätverket skall se ut. I en produktionskedja, t.ex., så kan det vara hur många lagerlokaler det skall finnas, vilken kapacitet de skall ha och vart de skall placeras. Den strategiska planeringen är viktig eftersom misstag i den ofta är svåra och dyra att rätta till i efterhand.

Den taktiska planeringen försöker att allokera och utnyttja resurserna i distributionsnätverk, detta sker vanligtvis över en dag eller en hel vecka. Detta så att ett företaget samtidigt kan uppnå ekonomiska och kundservicerelaterade mål. Exempel på taktisk planering kan vara schemaläggning av transporter för en viss tidsperiod.

Den operativa planeringen sker på en lokal nivå i en mycket dynamisk miljö. Den är kortsiktigt och innefattar bland annat. beslut så som att lägga om transporter medan de pågår.

## 1.2 Design av distributionsnätverk

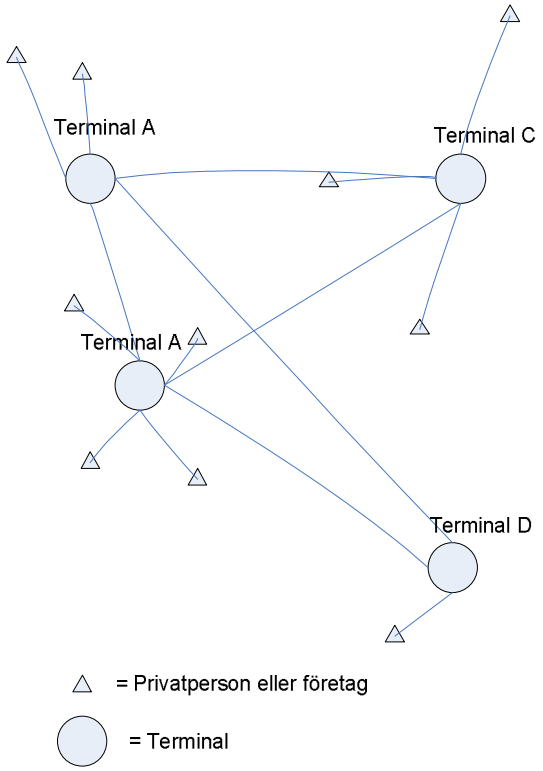
I design av distributionsnätverk, som tillhör den strategiska planeringen, gäller det att hitta de optimala placeringarna av komponenterna i systemet. Det kan t.ex. vara hur placering av fabriker och lagerlokaler ska ske för att minska transport- och hanteringskostnader. Design av distributionsnätverk är viktigt för att i ett väldesignat distributionsnätverk är det lättare att hålla nere transportkostnader och leverera varor snabbare. En dålig design kan leda till ett omständligt nätverk, vilket medför sämre service till kunder (Chan, Chung, 2004) . Olika distributionsnätverk har olika krav på design, t.ex. är ett vattenledningsnätverk statiskt i den meningen att det inte förändras så mycket efter det är byggt. När väl alla vattenledningar, pumpar och vattentorn är byggda är det kostsamt att förändra nätverket och därför är det viktigt att designen blir rätt från början. Ett postdistributionsnätverk är lite mer flexibelt på det sättet att transporterna mellan postterminaler kan ändras efter att det är byggt. Det finns mycket forskning kring design av distributionsnätverk eftersom det är ett verkligt problem

som många företag ställs inför när de t.ex. skall etablera sig på en marknad eller göra en omstrukturering av företaget. Det är också ett komplext problem då det sällan går att hitta en optimal design. Istället används ofta metaheuristiker för att hitta tillräckligt bra lösningar. Några exempel är Ponce de Leao och Matos (1999) som använder sig av simulated annealing och Chan och Chung (2004) som använder en genetisk algoritm. I ett väldesignat postdistributionsnätverk finns det större möjligheter att hitta bra transportlösningar än om postdistributionsnätverket är onödigt komplext eller bara dåligt designat.

## 1.3 Postdistributionsnätverk

Ett postdistributionsnätverk består av ett antal terminaler. Dessa terminaler samlar in post från privatpersoner och företag i dess närområde, den post som inte skall ut igen till terminalens närområden skickas till den terminalen som ligger närmast slutdestinationen. Figur 1 visar ett exempel på ett postdistributionsnätverk.

Det här arbetets fokus ligger på hur transporterna mellan terminalerna sker och inte hur posten kommer in och skickas ut till närområdena. Det som skiljer ett postdistributionsnätverk från de flesta andra distributionsnätverk är det att terminaler kan vara både producenter och konsumenter.




Figur 1: Exempel på ett postdistributionsnätverk

Transporter i postdistributionsnätverk är oerhört komplexa och väldigt svåra och tidskrävande att manuellt schemalägga och därför behövs det algoritmer som kan automatisera, förenkla och förbättra schemaläggningen av transporter. Det finns många faktorer som ligger utanför transporteringsproblemet som gör det svårare. Dessa faktorer är svåra eller omöjliga att kontrollera och är t.ex. arbetstider, störningar i vägnätet och fel i sorteringen vid terminalerna. Det finns också ofta flera olika transportsätt med olika för- och nackdelar. Vilka av faktorerna som skall tas med i beräkningarna är ett svårt problem. Ju fler relevanta faktorer, desto bättre bör schemaläggning bli på att komma fram till en lösning som fungerar bra i verkligheten

och inte bara i teorin. Men många faktorer medför också att problemet blir svårare att lösa. Det kan även vara svårt att avgöra vilka faktorer som är viktiga och hur de skall hanteras. Schemaläggningen av transporter kan ske både på en taktisk nivå då rutterna läggs för en viss tidsperiod och en operativ nivå då rutter kan ändras medan de håller på att köras.

## 1.4 Kommersiella alternativ

Det finns ett antal kommersiella program som löser distributionsproblem. Några exempel är RouteSmart[1], ILOG Dispatcher[2] och Direct Route[3] för att nämna några. De är dock ganska dyra och det kan kosta för en medelstor flotta på 50 fordon från $8000 till $60 000, med installationskostnader på $75 till $150 per timme.

[1] (www.routesmart.com)

[2] (www.ilog.com)

[3] (www.appianlogistics.com)

# 2 Bakgrund

Att frakta gods mellan olika platser är ett gammalt problem som uppstår i alla samhällen. Det är dock ett väldigt stort problem med många olika aspekter. För att begränsa problemet är detta arbete inriktad mot postdistributionsnätverk. För att bättre kunna få en övergripande bild av problemet går det att dela upp det i tre delar, strategisk, taktisk och operativ planering (Crainic T.G, 2002).

På den strategisk nivå gäller det att konstruera postdistributionsnätverk på ett sådant sätt så att det fyller sin funktion (att flytta post mellan terminaler) samtidigt som det kan utnyttjas effektivt. Det kan t.ex. vara hur många terminaler det skall finnas, vilka sorteringsmaskiner det skall finnas på terminalerna, vart terminalerna geografiskt skall placeras, etc. Det är önskvärt att kunna minimera antalet terminaler då de oftast är dyra i drift, men för få terminaler kan leda till ett dåligt och ineffektivt postdistributionsnätverk. Jablonsky och Lauber (1999) konstruerade en algoritm åt det tjeckiska postverket som hittar bra placeringar av terminaler och rätt sorts sorteringsutrustning till terminaler utifrån ett givet scenario.

På den taktiska nivån gäller det att utifrån ett befintligt postdistributionsnätverk hitta ett transportupplägg utifrån ett givet scenario. Ett transportupplägg är en möjlig lösning till problemet, d.v.s. ett antal transporter. Hur detta transportupplägg ser ut beror på scenariot och vilka krav som ställs på en lösning. Det viktigaste för de flesta postdistributionsnätverk är att all post skall komma fram till sin slutdestination. Om det finns en tid på när posten måste vara framme (vilket det nästan alltid finns) så måste den också uppfyllas. Det som prioriteras efter att posten måste komma fram (och i tid) brukar vara kostnaden. Det finns även andra tänkbara mål att optimera mot som t.ex. miljöpåverkan. Om problemet innefattar en egen flotta av fordon och arbetskraft ingår det i den taktiska planeringen att allokera och utnyttja flottan och arbetskraften på bästa sätt. Hollis et.al. (2005) skapade en algoritm som hittar bra lösningar på ett kombinerat transport- och fordonbemanningsproblem (multi depot simultaneous vehicle crew scheduling) åt det Australiensiska postverket.

Det finns många olika aspekter på detta problem, vilka som är relevanta beror på hur postdistributionsnätverket ser ut och omständigheterna runt detta. T.ex. om den som har postdistributionsnätverket har en egen flotta av fordon uppstår en rad nya problem, som t.ex. hur transporterna skall bemannas för att utnyttja personalen på bästa sätt och vart fordonen skall placeras efter dess sista körning. Vilka av dessa aspekter som inkluderas i problemet är olika från fall till fall.

Ett exempel på ett problem från den tyska posten är ”empty balancing”. I Tyskland skickas mer post från väst till öst än tvärtom vilket medför att det blir ett överskott av tomma containrar i östra Tyskland. Hur dessa tomma containrar ska omfördelas till nästa planeringsperiod är problemet.

Komplexiteten på problemet gör att en mjukvarubaserad lösning med fördel kan användas för att stödja planeringen. En sådan lösning brukar kallas DSS (Decsion Support System). Ett exempel på en DSS är ISLT (Information System for Letter Transportation) som skapades för att underlätta manuell planering av den tyska postens transporter. Systemet svarar på ”what if”-frågor som användaren ställer (Grunert and Sebastian, 2000).

Ett transportproblem som är välkänt och välstuderat är Vehicle Routing Problem. Det är också likt problemet att frakta brev i postdistributionsnätverk.

## 2.1 Vehicle Routing Problem

Vehicle Routing Problem (VRP) är ett välkänt problem för optimering av transporter i ett distributionsnätverk. Detta problem introducerades av Dantzig och Ramser (Dantzig G.B. och Ramser R.H., 1959) i slutet av femtiotalet och har undersökts och forskats i sedan dess. Det kan liknas vid en sammanslagning av Traveling Salesman Problem och Bin Packing Problem (Ralphs et al., 2001), där det gäller att frakta gods från ett centrallager till ett antal kunder via vägar som har olika kostnader. Dessa transporter sker med ett antal fordon. Varje rutt måste starta och sluta vid det centrala lagret och varje kund får bara besökas en gång av ett fordon. Problemet är att hitta de rutter som har den minsta totala kostnaden.

Det finns många olika varianter av VRP och de kan kombineras på många olika sätt. VRPTW (Vehicle Routing Problem with Time Windows) har ett ytterligare ett kriterium och det är att betjäning av en kund måste starta inom ett visst tidsintervall. I CVRP (Capacitated Vehicle Routing Problem) har alla fordon en uniform kapacitet, målet med CVRP är att hitta de rutter som har den minsta totala kostnaden samtidigt som mängden varor på varje enskild rutt inte får överstiga kapaciteten hos fordonen. VRPSP (Vehicle Routing Problem with Split Pickups) tillåter att en kund betjänas av mer än ett fordon. Vilket i sin tur tillåter att kunder kan ha ett större begär av varor än kapaciteten på fordonen. I VRPPD (Vehicle Routing Problem with Pick up and Delivering) kan kunder inte bara ta emot varor utan de kan också lämna ifrån sig varor till fordonen. I OVRP (Open Vehicle Routing Problem) måste fordonen inte återvända till starten efter de lämnat av varor. Alla dessa varianter är relaterade, mer eller mindre, till problemet att optimera transporter i postdistributionsnätverk.

De flesta av algoritmerna riktar in sig på batch-processing, d.v.s. alla transporter läggs från början och sedan ändras de aldrig. Men det finns även dynamiska algoritmer som hela tiden uppdaterar transporterna beroende på omständigheter runt problemet. Det kan t.ex. vara omständigheter på vägarna. En väg som var ett bra alternativ när transporten lades kan det ha blivit trafikstockning på i efterhand. En dynamisk algoritm kan anpassa sig till sådana situationer och lägga om transporten. Det kan också hända att nya transporter måste schemaläggas medan de ursprungliga transporterna håller på att köras. Dessa nya transporter måste på ett effektivt sätt kunna integreras med de gamla. Haghania och Jung (2005) presenterar en genetisk algoritm som löser en dynamisk VRPTW.

VRP är NP-Hard (Kallehauge et al., 2005), d.v.s. det finns ingen känd algoritm som kan lösa det optimalt på polynomial tid, vilket gör att det med dagens teknik är praktiskt omöjligt att lösa det exakt för större instanser av problemet. Detta gör att heuristiker ofta används för att hitta acceptabla lösningar.

## 2.2 Linjära program

Många olika algoritmer har applicerats på VRP. En del av dem hittar den optimala lösningen. Linjära heltalsprogram kan lösa ett VRP optimalt, med förutsättning att problemet är matematiskt formulerat. För att beskriva ett linjärt heltalsprogram behövs först en förklaring vad ett linjärt program är. Ett linjärt program är ett optimeringsproblem, där det gäller att maximera en funktion samtidigt som vissa villkor måste hålla. Ett linjärt program består normalt av tre delar.

- En linjär funktion som skall maximeras
- Ett antal invarianter
- Ett antal icke-negativa variabler

Ett linjärt program kan lösas på polynomial tid, med t.ex. Leonid Khachiyan algoritm som han introducerade 1979. Simplex algoritmen, som introducerades av George Dantzig 1947, har visserligen en komplexitet som i värsta fall är exponentiell men i praktiken har den en bättre prestanda än Leonids algoritm (Franklin, 1983).

Ett linjärt heltalsprogram liknar ett linjärt program med den skillnaden att variablerna måste vara heltal. En ytterligare variant är när det finns både variabler som måste vara heltal och variabler som inte behöver vara det. De brukar kallas blandade linjära heltalsprogram (mixed integer programming). De flesta verkliga problem kräver heltalslösningar. Ett linjärt heltalsprogram kan användas för att lösa VRP exakt. En matematisk modell av VRP skapas och därefter löses programmet med en algoritm. Det finns lite olika algoritmer för att lösa ett linjärt heltalsprogram, de vanligaste är branch-algoritmerna. De använder sig av divide and conquer, d.v.s. de delar upp problemet i mindre delar och löser delarna var för sig, för att lösa en sekvens av linjära problem som är förenklade versioner av det linjära heltalsproblemet. Branch and bound (Mitchell, 1999) är den enklaste. Branch and cut (Mitchell, 1999) och branch and price (Barnhart et al., 1996) är båda mer komplicerade och bättre än branch and bound, i den meningen att de kan undvika att utforska delar av trädet som inte innehåller någon lösning som är bättre än de lösningar som redan hittats.

## 2.3 Algoritmer som applicerats på VRP

Optimala algoritmer kan sällan användas på stora instanser av VRP på grund av deras höga komplexitet. Den största instansen av VRPTW som har lösts optimalt är 1000 kunder (Kallehauge et al., 2005). Men storleken på problemet behöver inte vara direkt kopplat till komplexiteten. Det finns testfall på några hundra kunder som visat sig svåra att lösa optimalt (Mitchell, 1999). Generellt sett går det inte hitta en optimal lösning inom rimlig tid på VRP som innehåller fler än 50 kunder (Oppen och Lokketangen, 2006). VRP är i sin ursprungliga form ganska simpel och det är sällan verkliga problem direkt kan göras om till ett VRP och lösas. Då många verkliga problem är större än 50 kunder måste andra lösningsmetoder användas. Eftersom VRP är ett problem som har stark anknytningen till många verkliga problem har det forskats mycket inom området och många olika lösningsmetoder har föreslagits. Några av de mer lyckade metoderna är tabu-search (Taillard et al., 1997), genetiska algoritmer (Alba et al., 2004) och ant colony system (Donati et al., 2002)

## 2.4 Genetiska algoritmer

Genetiska algoritmer är en typ av metaheuristisk algoritm som framgångsrikt har applicerats på VRP. Genetiska algoritmer tillhör de evolutionära algoritmerna. Dessa algoritmer försöker efterlikna den biologiska evolutionen. Genetiska algoritmer är en heuristisk sökmetod som är en analogi till naturligt urval och ”survival of the fittest”. De introducerades av Holland i mitten på sjuttio-talet (Holland, 1975). Genetiska algoritmer är inte optimala, utan de används för att hitta tillräckligt bra lösningar på optimeringsproblem, när optimala algoritmer inte kan användas då de har för hög komplexitet.

Vid början skapas en slumpmässig population av kandidatlösningar. Dessa kandidatlösningar utvärderas och tilldelas ett värde (fitness) beroende på hur bra lösningarna är. Sedan väljs två lösningar ur population. Det finns många olika urvalsmetoder, en vanlig metod är att ju högre fitness en kandidatlösning har desto högre chans har den att bli vald. Dessa två korsars och bildar två nya lösningar. Hur denna crossover sker beror på hur representationen ser ut. Vanligast är one-point

crossover som delar varje lösning i två delar, den första delen från den första lösningen kombineras med den andra delen från den andra lösningen och den andra delen från den första lösningen kombineras med den första delen av den andra lösningen. För att korsning skall vara meningsfullt bör lösningen kunna delas up i dellösningar. Det är inte alla problem som lätt kan delas up, t.ex. bestämning av vikter i ett ANN (Artificiellt Neuralt Nätverk). Det är ofta inte meningsfullt att ta två ANN och slå ihop dem eftersom det nya ANNet troligen kommer vara helt annorlunda. För en beskrivning av ANN se Callan (1999). Sedan muteras de två nya lösningarna och läggs till i populationen. Hur denna mutation sker beror på problemet och hur det är representerat. Därefter väljs två nya lösningar ut och de paras och muteras. Detta fortsätter tills en tillräckligt bra lösning har hittats eller ett antal förutbestämda generationer har passerat.

## 2.5 Hill climbing

Hill climbing är en lokal sökalgoritm, d.v.s. den söker efter bättre lösningar i närheten av där den befinner sig för tillfället. Den använder sig av två problemspecifika funktioner, en mutationsoperator som muterar lösningen och en målfunktion som evaluerar hur bra lösningen är. Den behöver även en lösning att starta ifrån. Den lösningen kan t.ex. vara tom, slumpas fram eller utgå från en redan existerande lösning.

Det enda som den håller reda på är lösningen den har för tillfället och därför har den små minneskrav. Hill climbing är inte optimal i det avseendet att det inte är säkert att den hittar den bästa lösningen, utan den kan lätt fastna i lokala optima.

Både mutationsoperatorn och målfunktionen måste anpassas efter problemet och den representation som används.

Hill climbing börjar med en startlösning som evalueras i målfunktionen, därefter muteras den med mutationsoperatorn. Den förändrade lösningen evalueras sedan i målfunktionen. Om den muterade lösningen är bättre än den gamla används den istället, om den inte är bättre kastas den bort. Sen sker en ny mutation på lösningen och den evalueras, om den är bättre används den, om inte så kastas den bort. Detta fortsätter till ett givet antal iterationer gjorts eller något annat stopkriterium är uppfyllt.

# 3 Problemdefinition

## 3.1 Introduktion

Att frakta gods är en viktig del av ekonomin. Det stödjer produktion, handel och konsumtionsaktiviteter genom att se till en effektiv och punktlig tillgänglighet av råmaterial och färdiga produkter. Transporter står för en stor del av kostnaden för en färdig produkt och är en stor del av ett lands nationella utgifter. Detta gör att konkurrensen bland fraktföretag är hög (Crainic, 2002).

Ett effektivt distribueringsnätverk kan innebära stora besparingar för ett företag. Trots detta är den fysiska distribueringen av gods ett av de minst utforskade områdena, vilket lämnar stort utrymme för förbättringar (Georgakopoulos och Mihiotis, 2004).

I postdistribueringsnätverk måste posten komma fram i tid samtidigt som kostnaden för transporterna måste hållas nere. Ett av de största problemen är balansen mellan en kort leveranstid och en låg kostnad. Dessa kriterier är konflikterande och gör planering av transporter svårt (Grunert and Sebastian, 2000). Ett sätt att hålla nere kostnaden är att samköra transporter. Samkörning är oftast helt nödvändigt för kostnaderna inte skall bli helt orimliga, men möjligheten att göra detta begränsas starkt av tidskraven på posten.

## 3.2 Problem

Problemet är att optimera transporter av brev mellan terminaler i postdistributionsnätverk. Optimeringen sker mot två mål i detta arbete, antal brev som inte transporteras och kostnaden. Detta gör problemet svårare att lösa, då målen måste värderas mot varandra. I detta arbete läggs störst vikt till att alla brev skall transporteras, det är först efter alla brev har kommit fram till sin slutdestination som en fokuserad optimering mot kostnaden sker. I postdistribuering vill man ofta minimera kostnaden för att transportera posten, samtidigt som den måste komma fram i tid. Tiden är inget mål som optimeras utan ett villkor som måste gälla för en giltig lösning.

## 3.3 Mål och delmål

Det här projektet är inriktat mot taktiskt planering i befintliga postdistributionsnätverk. Målet är att undersöka hur mycket heuristik kan förbättra algoritmer för optimering av transportupplägg i postdistributionsnätverk.

För att uppnå målet måste ett antal delmål uppfyllas:

1) Skapa en förenklad och generell simuleringsmodell utifrån en befintlig simuleringsmodell. Syftet med den förenklade modellen är att den skall gå snabbare att köra än den befintliga simuleringsmodellen, för att möjliggöra effektiv optimering.
2) Undersök olika optimeringsalgoritmer och representationer för att hitta de som passar till problemet, välj sedan en optimeringsalgoritm och en representation.
3) Använd ett planerat scenario på den förenklade modellen. Scenariot är en instans av ett postdistribueringsproblem och är nödvändig för att kunna göra en realistisk jämförelse. Konsekvenserna av att bara använda ett scenario mot den förenklade modellen är det blir svårare att påvisa att modellen är generell.

De slutsatser som dras från resultaten blir mer begränsande till just det specifika scenariot än om flera scenarion använts.

4) Jämför användningen av algoritmen på det planerade scenariot med och utan heuristik

# 4 Tillvägagångssätt

## 4.1 Förenklad Modell

För att ta fram en förenklad modell som kan representera postdistributionsnätverk studeras en befintlig modell. Denna befintliga modell ligger till grund för den förenklade modellen. Simuleringsmodellen som studeras är framtagen för Posten Sverige och heter PRiTSi (Wrangtorp, 2006). Arbetet med PRiTSi startade hösten 2001 och utvecklingen av den har skett kontinuerligt. Mycket arbete har lagts ner i PRiTSi och därför antas att den är korrekt.

Anledningen till att använda en befintlig modell som utgångspunkt är att det går att använda mycket av det arbetet som har lagts ner på PRiTSi i den förenklade modellen, det går på så vis att få en bra modell på kort tid. Samtidigt som den blir mer realistisk än om den byggts från grunden. Nackdelen med denna metod är dock risken att modellen inte blir så generell och att det kan bli svårt att applicera den förenklade modellen på andra postdistributionsnätverk. Anledning till att inte optimera mot PRiTSi direkt är att den är en stor och komplex modell och en simuleringsomgång tar för lång tid. Syftet med den förenklade modellen är att en simuleringsomgång skall gå mycket snabbare.

## 4.2 Representation och algoritmer

Den befintliga modellen studeras för att hitta en representation som passar till problemet. Detta för att den kan innehålla information som kan användas för att skapa en bra representation. Representationen bör passa den algoritm som väljs. Problemet med att välja en representation utifrån den befintliga simuleringsmodellen kan vara att representationen kanske anpassas för mycket till PRiTSi så den inte passar för andra postdistributionsnätverk.

För att hitta lämpliga algoritmer som passar till problemet studeras befintlig litteratur. Den litteratur som studeras är främst vetenskapliga tidskrifter om metaheuristiker på VRP. För att välja algoritm kommer nackdelar och fördelar jämföras för att hitta den bästa algoritmen.

## 4.3 Planerat scenario

Ett planerat scenario från Posten Sverige appliceras på den förenklade modellen för att ha ett realistiskt scenario som algoritmerna kan användas på. Anledningen till att använda ett scenario från PRiTSi är den förenklade modellen är framtagen från PRiTSi så ett scenario därifrån bör lätt passa in i den förenklade modellen. Ett planerat scenario från ett annat postdistributionsnätverk hade antagligen inte passat in på den förenklade modellen utan modifieringar då de flesta verkliga postdistributionsnätverk skiljer sig åt. Problemet med att bara använda ett planerat scenario är att det är svårt att påvisa att den förenklade modellen är generell.

## 4.4 Applicera algoritmen på den förenklade modellen

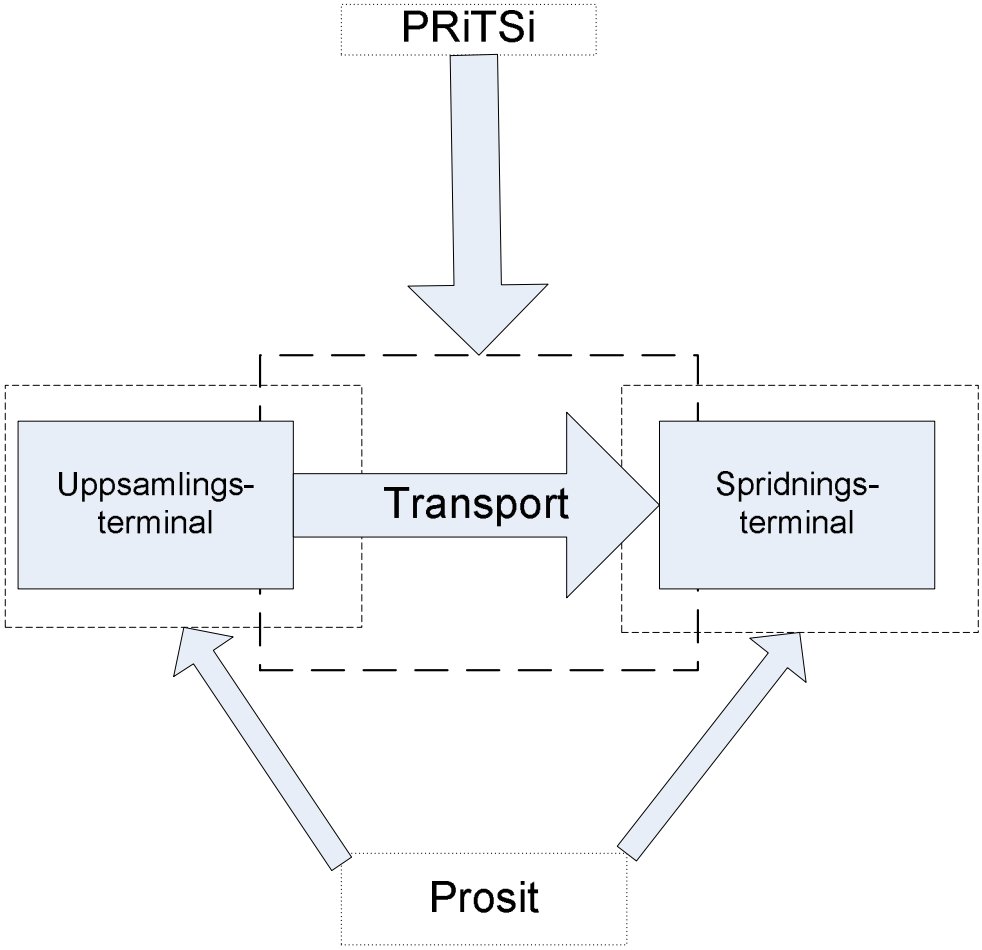
För att undersöka om heuristik kan användas för att förbättra vald algoritm implementeras och appliceras den på den förenklade modellen med det planerade scenariot. All implementation sker i programspråket C++/CL.

# 5 Simuleringsmodell

## 5.1 Introduktion

Den simuleringsmodellen som ligger till grund för den förenklade modellen är PRiTSi (Posten RiksTransporter Simulering) och är framtagen av företaget Establish för Posten Sverige[4]. Den är en av två simuleringsmodeller för att simulera flödet av post i Postens postdistributionsnätverk.

Figur 2 beskriver hur de två simuleringsmodellerna är relaterade. Prosit simulerar hur posten hanteras och sorteras på uppsamlings- och spridningsterminaler. Det som blir utdata från Prosit blir indata till PRiTSi. PRiTSi simulerar hur transporter sker mellan terminaler.




Figur 2: Postens två simuleringsmodeller och deras relation (Ekberg och Stablum, 2006).

## 5.2 PRiTSi

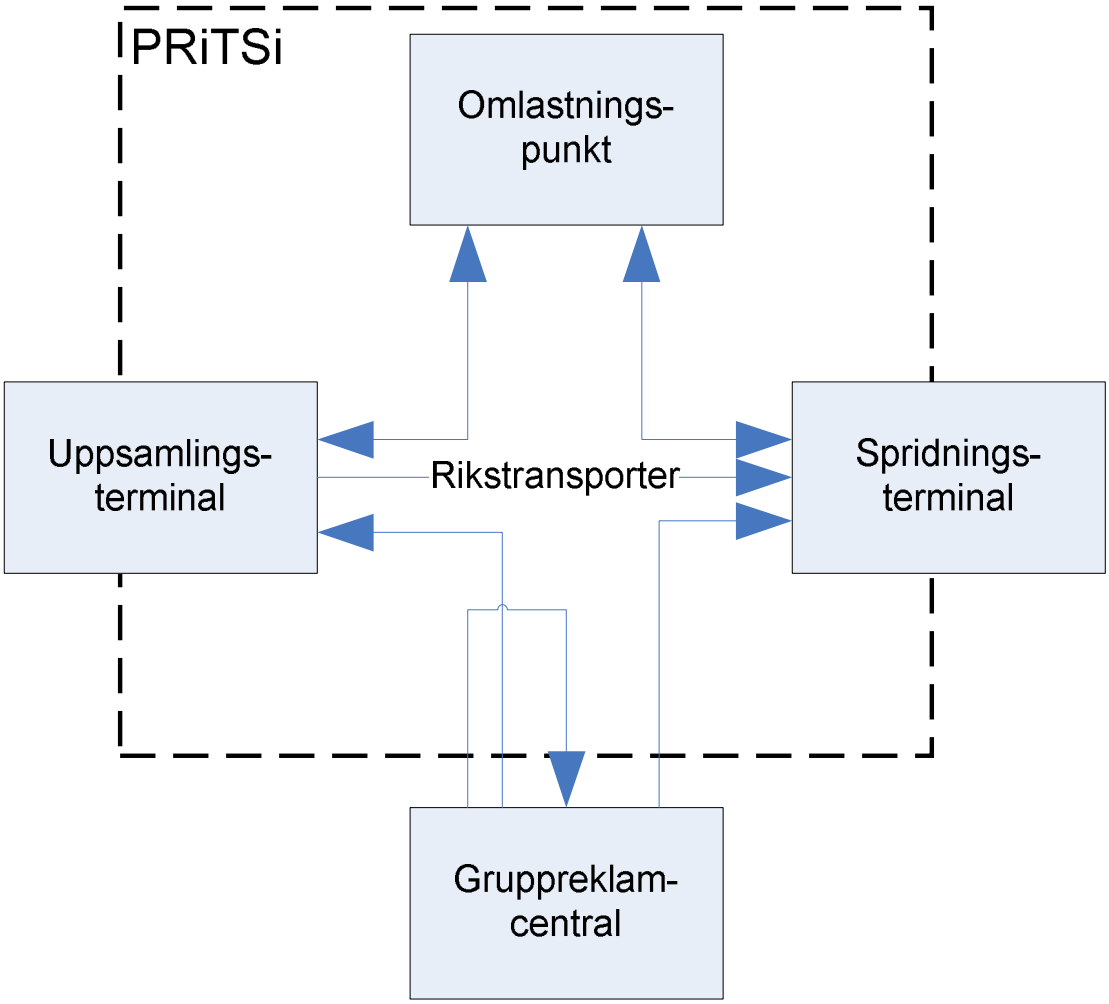
PRiTSi är en nätsimuleringsmodell för transporter mellan terminaler i Postens postdistributionsnätverk. Den används av Posten bl.a. för att är att simulera störningar i riksnätet, olika transportupplägg, byte av transportupplägg i en viss relation, ny terminalsstruktur ur transportsynpunkt samt beredskapsplaner.

[4] http://www.posten.se

Utifrån PRiTSi skapas en generell modell som kan användas för att simulera transporter mellan terminaler. Huvudanledningarna för att inte använda PRiTSi direkt är att PRiTSi är special designat för Posten Sverige och det är osannolikt att den kan användas på andra postdistribueringsnätverk direkt utan modifieringar. En annan anledning är att en simuleringsomgång är tidskrävande. På en vanlig PC tar det flera minuter med ett realistiskt transportupplägg. Ur ett optimeringsperspektiv är detta för lång tid eftersom många körningar behöver göras. En förenklad modell kan köras flera gånger per sekund vilket ger en större chans att kunna hitta ett bra transportupplägg.

För att bättre kunna förstå hur den förenklade modellen skapas ifrån PRiTSi beskrivs PRiTSi övergripande. PRiTSi är en stor och komplex modell och den innehåller väldigt mycket information. En fullständig beskrivning av PRiTSi är därför inte rimligt och på grund av det så utelämnas mycket information. Allting som inte beskrivs antas vara ointressant för den förenklade modellen. Dock förklaras viktiga förenklingar av PRiTSi.

Figur 3 beskriver hur transporter inom PRiTSi kan ske. Transporter kan ske mellan uppsamlings- och spridningsterminaler direkt eller de kan ske via en omlastningspunkt. Transporter från en gruppreklamcentral kan gå till en uppsamlingsterminal, spridningsterminal eller till en annan gruppreklamcentral. Ingen simulering av den interna verksamheten i en gruppreklamcentral sker i PRiTSi.

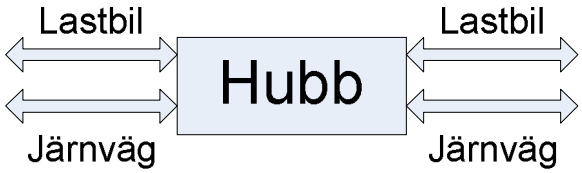



Figur 3: PRiTSi (Ekberg och Stablum, 2006).

## 5.3 Terminologi

En terminal är en fysisk punkt där sorteringsverksamhet bedrivs. En terminal kan båda vara en uppsamlings- och spridningsterminal. En uppsamlingsterminal samlar in posten ifrån sitt närområde och sorterar den på terminalnivå. I en spridningsterminal spridningssorteras posten till kunder i terminalens ansvarsområde. En terminal fungerar alltså både som en konsument och producent av post.

En omlastningspunkt kategori 1 simulerar verksamheten vid en omlastningspunkt av hubbtyp. En transport bryts på en omlastningspunkt kategori 1 och en form av hantering sker. Figur 4 visar hur en transport av typen lastbil eller järnväg kommer till en hubb och fortsätter därifrån med lastbil eller järnväg. Inga transporter kan specificeras att sluta i en omlastningspunkt kategori 1, sträckan kan dock sluta i en. Detta därför att alla transporter måste specificera vart posten kommer ifrån och vart posten skall (slutdestinationen) och ingen post kan ha sin slutdestination i en omlastningspunkt kategori 1, då detta enbart är en hubb som samlar ihop och förmedlar post vidare. Om en sträcka slutar i en omlastningspunkt kategori 1 innebär detta att en ytterligare transport kommer att krävas för att ta posten till sin slutdestination.




Figur 4: Hubb

I modellen finns det också omlastningspunkt kategori 2, omlastningspunkt kategori 3 och spetsbyte kategori 4. Dessa omlastningspunkter är snarlika kategori 1 omlastningspunkter och beskrivs inte närmare då de inte används i den förenklade modellen.

Det finns tre stycken olika huvudtransportsätt, de är lastbil, tåg och flyg. De i sin tur har lite olika transportsätt, t.ex. lastbil, lastbil med släp, tåg med en vagn, tåg med två vagnar, litet flygplan, stort flygplan o.s.v.

En delsträcka är två punkter som är direkt förbundna med varandra. Punkterna kan vara terminaler eller omlastningspunkter. Till varje delsträcka finns ett transportsätt associerat samt den tid det tar att använda delsträckan.

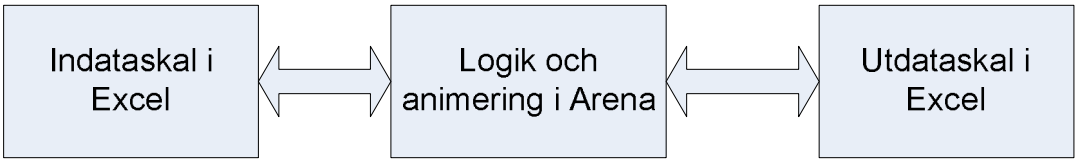
En sträcka består av en eller flera delsträckor. En sträcka börjar alltid på en terminal eller en omlastningspunkt kategori 1 och slutar alltid på en terminal. Sista delsträcka kan dock sluta på en omlastningspunkt kategori 1 och om detta sker måste en ny transport definieras som tar hand om den sista transporten.

En transport består av en sträcka och en starttid.

Ett transportupplägg består av ett antal transporter. Transportupplägget är alla transporter som kommer att köras i simuleringen.

## 5.4 Modellens uppbyggnad

Figur 5 beskriver hur modellen är uppbyggd. Den består av två delar, Arena och skalet.




Figur 5: Kopplingen mellan skalet och Arena

Arena är en modelleringsmjukvara som sköter logiken och animeringen av simuleringen. Den hämtar indatan till simuleringen från skalet. När simuleringen är klar skrivs resultaten från simuleringen till skalet.

Skalet är ett Excelark, bestående av flera blad, som hanterar in- och utdatan till Arena.

Ur ett optimeringsperspektiv finns det två typer av indata till Arena. Den ena typen av indata är villkor som t.ex. vilka brevterminaler som finns, hur stora avstånd är mellan brevterminaler, hur mycket post som finns på varje terminal och vart den posten skall. Denna typ av indata ändras inte av optimeringen utan är fast.

Den andra typen av indata är variabler, det är de som ändras av optimeringen. Det kan vara vilka sträckor som skall köras och vilken typ av fordon som skall användas.

Posten är indelad i två stycken typer, de är A-post och B-post. B-post bortses helt ifrån. Därför kommer A-post hädanefter i detta arbete refereras till som post.

Det finns tre stycken olika postsorter, de är C4, C5 och klump. Dessa kan sedan delas in ytterligare i maskinell C4, manuell C4, maskinell C5, manuell C5, maskinell klump och manuell klump. Maskinell betyder att de kan sorteras i en maskin på en spridningsterminal och manuellt betyder att de måste sorteras manuellt.

För att ta reda på hur mycket post som skall transporteras och vart den skall måste ett antal steg genomföras. All information som behövs finns i bladen ”Alternativt inflöde”, ”Struktur, terminaler”, ”Volymuppdelning”, ”Pnr-område” och ”Lastområden”.

Först hämtas information ifrån bladet ”Alternativt inflöde” (figur 6). I den finns det hur mycket post som finns på en terminal och vid vilken tidpunkt den finns tillgänglig.

| Från Terminal | Tillgängligt för trp | | Volym | | Kundsort. A-post | |
|---|---|---|---|---|---|---|
| Sortera | Kl | Dag | A-post | B-post | Volym | K.sort. Format |
| Alvesta | 20:30 | F1 | 183 937 | | | |
| Alvesta | 21:35 | F1 | 103 693 | | | |
| Göteborg | 20:00 | F1 | 387 736 | | | |
| Göteborg | 20:30 | F1 | 90 190 | | | |
| Göteborg | 21:35 | F1 | 160 216 | | | |

Figur 6: Alternativt inflöde

Sedan hämtas till vilka terminaler som varje postnummerområde hör, detta hämtas ifrån bladet ”Lastområden” (figur 7).

| Lastningsområden | LO 1 | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Spr-terminal / Ingående. PO | Från PO | Till PO | Från PO | Till PO | Från PO | Till PO | Från PO | Till PO |
| Malmö | 20 | 21 | | | | | | |
| Alvesta | 30 | 32 | | | | | | |
| Göteborg | 40 | 41 | | | | | | |
| Nässjö | 52 | 54 | | | | | | |

Figur 7: Lastområden

Därefter hämtas hur många procent som är C4, C5 och klump av posten som finns i ”Alternativt inflöde” samt hur många procent av C4, C5 respektive klump som kan sorteras maskinellt. Den informationen finns i ”Struktur, terminaler” (figur 8).

| Terminal | Formatfördelning A (%) | | | Formatfördelning B (%) | | | Andel till maskin A (%) | | |
|---|---|---|---|---|---|---|---|---|---|
| | C5 | C4 | Klump | C5 | C4 | Klump | C5 | C4 | Klump |
| Malmö | 78% | 17% | 5% | 65% | 23% | 12% | 92% | 73% | 88% |
| Alvesta | 80% | 16% | 4% | 84% | 16% | 0% | 90% | 76% | 0% |
| Göteborg | 80% | 15% | 5% | 71% | 21% | 8% | 88% | 75% | 90% |
| Nässjö | 80% | 16% | 4% | 72% | 27% | 0% | 92% | 79% | 0% |
| Norrköping | 79% | 16% | 5% | 64% | 34% | 2% | 81% | 61% | 0% |

Figur 8: Struktur, Terminaler

Sedan hämtas informationen om hur många procent av de olika postsorterna som finns i varje pnr-område från ”Volymuppdelning, Pnr-område” (figur 9).

| Ups Terminal | Summa | Pnr-områden | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| A: Mask C5 | | | | | | | | |
| Malmö | 100,0% | 2,4% | 3,4% | 1,1% | 0,8% | 0,6% | 0,3% | 0,9% |
| Alvesta | 100,0% | 2,4% | 3,3% | 1,2% | 1,1% | 0,9% | 0,4% | 1,0% |
| Göteborg | 100,0% | 2,3% | 3,4% | 0,9% | 0,7% | 0,5% | 0,3% | 0,8% |
| Nässjö | 100,0% | 2,8% | 3,3% | 0,9% | 0,8% | 0,6% | 0,3% | 0,9% |

Figur 9: Volymuppdelning Pnr-område

När allt det är gjort finns all information som krävs för att räkna ut hur mycket post som finns på varje terminal, när den finns tillgänglig och vart den skall.

Utdatan från simuleringen skrivs till speciella utdatablad i skalet. De viktigaste är kostnaden samt hur många brev som inte kommit fram. Det finns dock många andra parametrar som vilka transporter som kördes, fyllnadsgraden hos transporter, när transporterna startade, hur mycket post som blev kvar på varje terminal, hur mycket post som skickades från varje terminal, när posten kom fram, miljöbelastning o.s.v. Figur 10 visar några exempel på utdata.

| Produkt | Transport slag | Antal försänd. | Snittförs Transportkm | Tot Kostnad Alt 1 SEK | Miljöbel. |
|---|---|---|---|---|---|
| A:C5 | Flyg | 334 160 | 0 | 790 341 | 206 |
| | Väg | 2 141 511 | 76 | 170 641 | 79 360 |
| | Tåg | 621 142 | 288 | 261 572 | 0 |
| | *Tot* | *3 096 813* | *182* | *Tot: 1 222 554* | *Tot: 79 566* |

Figur 10: Transportkostnader

För att få fram kostnaden för en sträcka krävs information från bladet ”Transportslag”, ”Avståndsmatris väg” och ”Avståndsmatris tåg”.

I bladet ”Transportslag” (figur 11) finns det specificerat hur mycket de olika transportsätten kostar. Varje transportsätt har fyra stycken olika kostnader, de är kostnad per timme, kombinations-kostnad per timme, kombinations-kostnad per km och kostnad per km. Vilka av de olika kostnaderna som gäller beror på sträckans längd. Det finns två stycken brytpunkter som bestämmer vilka kostnader som gäller. Innan första brytpunkten gäller enbart kostnad per timme, mellan första och andra brytpunkten gäller kombinations-kostnad per timme och kombinations-kostnad per km och efter andra brytpunkten gäller kostnad per km.

| Fordonstyp 67 | Variant | Animerings-bild | Kajtyp | Kostnad per timme | Brytpunkt1 km | Kombinations-kostnad per timme | per km | Brytpunkt2 km | Kostnad per km |
|---|---|---|---|---|---|---|---|---|---|
| Flyg 16 | Flyg | Stort flygplan | | 100000 | | | | | |
| Flyg 6 | Flyg | Litet flygplan | | 35000 | | | | | |
| Lastbil+släp | Väg | Tung lb+släp | Fjärrkaj | 1480 | 50 | 899 | 23,15 | 99 | 11,9 |
| Lastbil | Väg | Tung lastbil | Dist.kaj Hög | 1237 | 50 | 664 | 16,33 | 99 | 10,07 |
| Lätt lastbil | Väg | Lätt lastbil | Dist.kaj Låg | 0 | 0 | 1280 | 20,78 | 99 | 10,67 |
| Skåpbil | Väg | Lätt lastbil | Dist.kaj Låg | 2200 | 50 | 655 | 10,91 | 99 | 9,63 |

Figur 11: Transportslag

I bladen ”Avståndsmatris väg” (figur 12) och ”Avståndsmatris tåg” (den är uppbyggd på samma sätt som ”Avståndsmatris väg”) finns avstånden mellan de olika terminalerna i vägnätet respektive tågnätet.

| Från | Till Malmö | Till Alvesta | Till Göteborg | Till Nässjö |
|---|---|---|---|---|
| Malmö | | 190 | 274 | 297 |
| Alvesta | 190 | | 220 | 107 |
| Göteborg | 274 | 220 | | 192 |
| Nässjö | 297 | 107 | 192 | |

Figur 12: Avståndmatris väg

Med information från de olika bladen kan sedan kostnaden för en sträcka räknas ut.

För att få fram den maximala volymen för en sträcka krävs information från bladen ”Transportslag” och ”Lastbärare”.

Det finns ett antal olika lastbärare (figur 13), de finns specificerade i bladet ”Lastbärare”. En lastbärare är en fysisk enhet som används för att förvara och transportera post. Det kan t.ex. vara en lådvagn, låda, säck eller ett värdeskåp. Olika lastbärare kan innehålla olika typer av post och olika mycket av varje postsort. T.ex. så kan en säck bara ta klump post medan en BBH kan ta alla tre sorter.

| Lastbärare | Tara-vikt (kg) | C5 | C4 | Klump |
|---|---|---|---|---|
| Igloo | 145,00 | | | |
| Låda | 1,40 | 350 | 70 | |
| Lådvagn | 45,00 | 7 000 | 1 400 | |
| Säck | 0,20 | | | 45 |
| BBH | 78,00 | 5 250 | 1 050 | 180 |

Figur 13: Lastbärare

I bladet ”Transportslag” (figur 14) finns information om vilken typ av lastbärare och hur många av denna typ som finns tillgängliga till varje transportsätt.

| Fordonstyp 67 | Lastutrymme 1 Volym | Enhet | Lastutrymme 2 Volym | Enhet |
|---|---|---|---|---|
| Flyg 16 | 16 | Igloo | | |
| Flyg 6 | 30 | BBH | 40 | Säck |
| Lastbil+släp | 66 | BBH | | |
| Lastbil | 26 | BBH | | |
| Lätt lastbil | 6 | BBH | | |

Figur 14: Transportslag

Med informationen från de två bladen kan den maximala volymen C4, C5 och klump räknas ut för varje transport.

Om varje transport bara skulle ha med sig post från en terminal till en annan terminal hade det gått åt väldigt många transporter och dessa transporter hade fått köra nästan tomma. För att bättre kunna utnyttja transporternas kapacitet samlastas posten. Om t.ex. en transport går från terminal A till terminal C via terminal B kan den posten

som skall från terminal A till terminal C samlastas med den post som skall från terminal A till terminal B. Det är dock inte alltid det är möjligt att samlasta post, posten är kanske inte tillgänglig vid samma tidpunkt. Det kan också vara så att post som skall till två terminaler som ligger långtifrån varandra inte kan samlastas då transporten till terminalen längre bort måste åka tidigare för att den måste ha en viss tid på sig att sortera posten.

Kostnaden för en transport som samkörs är kostnaden för den transport som de andra går med. Det är alltså kostnadsfritt för de andra transporterna.

Att räkna ut volymen för transporter som samlastas är dock mer problematisk. Detta eftersom olika sträckor har olika transportsätt och olika transportsätt tar olika volymer med C4, C5 och klump post. Den maximala volymen hos en transport beror på hur mycket av de olika postsorterna som transporterna innan har tagit med sig. Om en transport är tillagd för att transportera C4 men transporterna innan har förbrukat all C4 plats men det finns gott om plats för klump post blir den transporten onödigt. Detta löses genom att dela in postsorterna i olika flöden och sätta olika prioriteter på flödena. När posten sedan skall lastas vid terminaler lastas postsorterna i den ordning som flödena är prioriterade.

I bladen ”Avgående trpt, Term och GrmC” (figur 16) och ”Avgående trpt, Oml.Kategori 1” (den är uppbyggd på samma sätt som ”Avgående trpt, Term och GrmC”) specificeras alla transporter som skall köras. En transport från en terminal till en annan skall läggas till i bladet ”Avgående trpt, Term och GrmC” och en transport ifrån en omlastnings punkt kategori 1 till en terminal skall läggas till i bladet ”Avgående trpt, Oml.Kategori 1”.

För att lägga till en transport i något av bladen måste sträckan finnas specificerad i bladet ”Nätverk” (figur 15). I ”Nätverk” finns alla sträckor som finns tillgängliga specificerade. Där finns också vilken transportsätt transporten använder sig av. ”Nätverk” innehåller även information om hur många delsträckor som sträckan innehåller.

| Från<br>Sök | Till | Färdsträcka typ | Delsteg<br>nr | Leg |
|---|---|---|---|---|
| Alvesta | Umeå | 1 | 1 | Alvesta-Umeå Lastbil+släp |
| Alvesta | Umeå | 2 | 1 | Alvesta-Tomteboda 2 Gbl |
| Alvesta | Umeå | 2 | 2 | Tomteboda-Sundsvall godskaj 10 Gbl |
| Alvesta | Umeå | 2 | 3 | Sundsvall godskaj-Umeå Lastbil+släp |

Figur 15: Nätverk

När en sträcka har valts som skall köras och den finns i ”Nätverk” är den redo att läggas till i ”Avgående trpt, Term och GrmC” (figur 16) eller ”Avgående trpt, Oml.Kategori 1”. Det som skall skrivas in i bladet är, frånterminalen, tillterminalen, vilket transportsätt som skall användas och när den skall avgå.

| Från<br>Terminal | Färdsträcka<br>typ | Till<br>Terminal | Avgångs<br>tid (hh:mm) | Dagar | Går |
|---|---|---|---|---|---|
| Nässjö | 4 | Karlstad | 8:30 | 1234 | |
| Norrköping | 2 | Västerås | X | | 10018 |
| Nässjö | 1 | Norrköping | X | | 10018 |
| Nässjö | 6 | Västerås | X | | 10018 |

Figur 16: Avgående trpt, Term och GrmC

Om en transport skall samköras behöver det specificeras vilken transport den skall gå med. Ingen tid behöver dock specificeras, då det hämtas från transporten den går med.

I bladet ”Samlastning” (figur 17) finns information som behövs för att ta reda på om transporterna kan samlastas. Där anges uppsamlingsterminalen, spridningsterminalen, vilket transportsätt och mellan vilken tid samlastningen är giltig.

| Ups-terminal eller Kat 1 hubb | Spr-terminal | Tp-slag | Split vid hubb | Villkor fyllnadsgrad | | Intervall avg.tid | |
|---|---|---|---|---|---|---|---|
| | | | | min | max | Start | Slut |
| Alvesta | Uppsala | Flyg | | | | 15:00 | 24:00 |
| Alvesta | Uppsala | Flyg | | | | 15:00 | 24:00 |
| Alvesta | Uppsala | Tåg | | | | 15:00 | 24:00 |
| Alvesta | Västerås | Flyg | | | | 15:00 | 24:00 |

Figur 17: Samlastning

# 6 Förenklad modell

## 6.1 Introduktion

Den förenklade modellens syfte är att simulera hur ett postdistributionsnätverk fungerar, fast mycket snabbare än PRiTSi. En simuleringsomgång i PRiTSi tar ett antal minuter. Detta är ofta en för lång tid vid optimeringar då simuleringen behöver köras många gånger. Så istället för att använda PRiTSi direkt kommer den förenklade modellen att användas.

Problemet är att transportera unika varor mellan noder. Dessa noder kan vara både producenter och konsumenter av varor. Noderna är sammankopplade med kanter, där varje kant har en kostnad, tid och en kapacitet. Det kan finnas flera kanter mellan två noder. Detta problem liknar det kända problemet VRP men skiljer sig på ett antal sätt. I VRP finns det bara en väg mellan två kunder. Den kanske största skillnaden är att i VRP gäller det att dela ut varor till kunder. Dessa varor är oskiljbara, det spelar ingen roll för kunderna vilka varor de får eftersom alla varor är likadana. I det här problemet så är alla varor unika och har en destination. Även målen skiljer sig lite. I det här problemet är det varan (post) som skall transporteras till sin slutstation, medan det i VRP är kunder som skall tillfredsställas med hjälp av varor.

Denna modell skapas utifrån PRiTSi. Förhoppningen är att denna förenklade modell skall beskriva hur verkligheten ser ut för många postdistributionsföretag. Risken är dock att den passar bäst till Posten Sverige. Om det går att hitta och optimera lösningar till den förenklade modellen så är förhoppningen att det går att applicera de lösningarna på PRiTSi och på så sätt kunna effektivisera postdistributionsnätverket, med antagandet att PRiTSi är korrekt. Då optimeringen sker mot kostnaden skulle detta kunna medföra besparingar för företaget.

## 6.2 Formell beskrivning

Nedan följer en mer formell beskrivning av problemet. Denna beskrivning är till för att lättare kunna förstå och implementera modellen.

Låt $G = (V, E)$ vara en riktad multigraf som består av $V = \{v_1, v_2, ...v_n\}$ som är mängden med alla hörn, $E = \{e_1, e_2, ...e_n\}$ som är mängden med alla kanter samt en funktion f från $E$ till $\{(u, v) \mid u, v \in V, u \neq v\}$ som anger mellan vilka hörn en kant går. Kanterna $e_1$ och $e_2$ är multipla kanter om $f(e_1) = f(e_2)$. Låt varje hörn motsvara en terminal och varje kant motsvara en väg. Låt *k* vara en kostnadsfunktion från *E* till *N*, *t* en tidsfunktion från *E* till *N* och *l* en kapacitetsfunktion från *E* till *N* där *N* är mängden av naturliga tal.

Låt $M$ vara en matris där $m_{ij}(n, t)$ innehåller $n_{ij}$ som är antalet brev som skall transporteras från terminal $v_i$ till terminal $v_j$ och $t_{ij}$ som är vid vilken tidpunkt de finns tillgängliga.

Låt $S = \{(e_1, e_2, ..., e_n) \mid e_1, e_2, ..., e_n \in E\}$ vara en sträcka som är en lista av kanter.

Låt $T = \{\{s_1, s_2, ..., s_n\} \mid s_1, s_2, ..., s_n \in S\}$ vara en transporttyp som är en mängd med sträckor. Varje sträckas lista med kanter måste vara en delmängd av den längsta sträckans lista med kanter. Detta för att sträckorna i en transporttyp samlastas och för att kunna göra detta måste de använda samma kanter.

Låt $R = \{a, t \in T\}$ vara en transport där $a$ är tiden då transporten startas och $t$ är transporttypen som används.

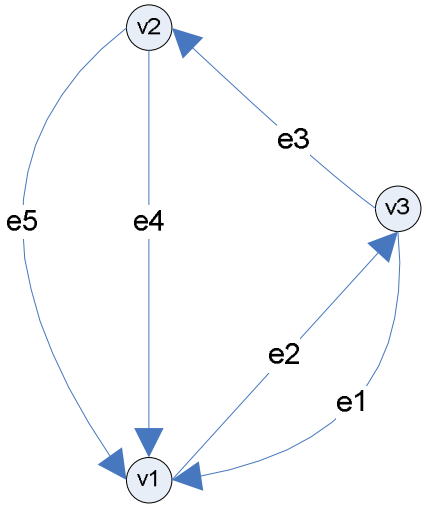
Låt $K = \{(r_1, r_2, ..., r_n) \mid r_1, r_2, ..., _n\ r \in R\}$ vara en kandidatlösning till problemet som är en lista med transporter.

I starttillståndet $t = 0$ är matrisen $M_0$ fylld med alla brev som skall distribueras. När en transport, vid tid *t*, av brev är klar skapas en ny matris $M_{t-1} \rightarrow M_t$. Sluttillståndet är när $M_x$ är en nollmatris, då har alla brev kommit fram till sin slutstation.

Det finns även en deadline *d* då alla brev måste ha kommit fram till sin slutdestination. Matrisen $M_d$ ska alltså vara en nollmatris.

Problemet är att hitta mängden av transporter som tar problemet från starttillståndet till sluttillståndet med den minsta totala kostnaden och inom givna tidsbegränsningar.

Ett exempel på ett problem med tre noder visas nedan. Figur 18 visar hur grafrepresentation ser ut.




Figur 18: En grafrepresentation av ett problem.

Det finns fem stycken kanter och deras kostnad, kapacitet och tid visas i figur 19.

| Fränterminal | Destinationsterminal | Kostnad | Kapacitet | Tid |
|---|---|---|---|---|
| v1 | v3 | 15 000 | 10 000 | 10 minuter |
| v2 | v1 | 20 000 | 30 000 | 40 minuter |
| v2 | v1 | 10 000 | 15 000 | 20 minuter |
| v3 | v1 | 5 000 | 5 000 | 5 minuter |
| v3 | v2 | 10 000 | 15 000 | 35 minuter |

Figur 19: Kostnaden, kapaciteten och tiden för kanterna i figur 18.

Brevmatrisen till exemplet visas i figur 20. Den innehåller hur mycket post det finns på varje terminal var den posten skall och vid vilken tidpunkt den finns tillgänglig.

| | v1 | V2 | v3 |
|---|---|---|---|
| v1 | | 60 000, 19:00 | 40 000, 19:00 |
| v2 | 12 000, 18:00 | | 7 000, 20:00 |
| v3 | 5000, 21:00 | 20 000, 21:00 | |

Figur 20: Ett exempel på hur en brevmatris kan se ut. Radnumret motsvarar på vilken terminal posten finns och kolumnnumret till vilken terminal posten skall.

Det finns tre sträckor och de visas i figur 21.

| Sträcka | Lista med kanter |
|---|---|
| $s_1$ | e2 och e3 |
| $s_2$ | e4 och e2 |
| $s_3$ | e1 |

Figur 21: Exempel på sträckor.

Det finns tre transporttyper, en för varje sträcka. De är $t_1 = \{s_1\}$, $t_2 = \{s_2\}$ och $t_3 = \{s_3\}$. Detta exempel tar inte upp samlastning, det är därför det bara finns en sträcka i varje transporttyp. Samlastning beskrivs mer noggrant längre fram i rapporten.

Det finns också en deadline vid 23:00, då skall all post vara framme.

Transporttyperna motsvaras av sträckor som bestämmer mellan vilka terminaler post kan transporteras. Så i detta exempel kan endast post från v1 till v3, från v3 till v2 och från v2 till v1 transporteras. En transport byggs upp av en transporttyp och en avgångstid. En transport är det som är dynamiskt, d.v.s. det är det enda som optimeringsalgoritmen använder sig av för att bygga upp en kandidatlösning. Den kan t.ex. inte skapa en ny kant eller ett nytt transportsätt, men den kan skapa en ny transport genom att ta en transporttyp och associera en avgångstid till den.

I exemplet finns en kandidatlösning som består av transporterna i figur 22. Det är kandidatlösningen som bestämmer hur och när posten skall transporteras och i detta exempel löser inte kandidatlösningen problemet då den inte transportera alla post i brevmatrisen före deadline.

| Transporttyp | Avgångstid |
|---|---|
| $t_1$ | 22:00 |
| $t_2$ | 23:00 |
| $t_3$ | 20:00 |

Figur 22: Exempel på transporter

## 6.3 Representation och datastrukturer

Nedan följer en mer detaljerad beskrivning av begreppen som introducerades i föregående avsnitt.

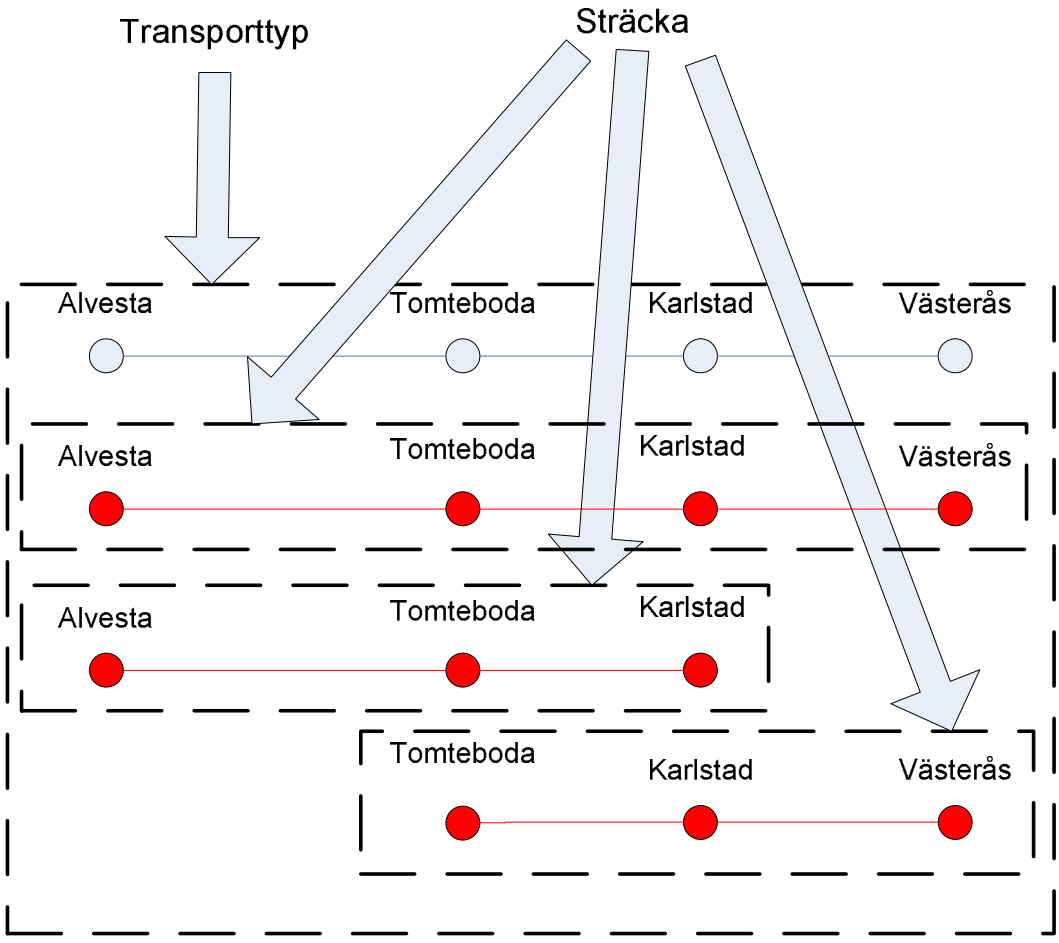
Brevmatrisen är en matris där radnumret motsvarar på vilken terminal posten finns och kolumnnumret motsvarar till vilken terminal posten skall. Varje element innehåller information om hur mycket post det finns och vid vilken tidpunkt de finns tillgängliga.

En delsträcka består av den punkt där den startar, den punkt där den slutar, hur lång tid det tar att använda den, hur mycket det kostar och hur mycket volym som maximalt får plats på den.

En sträcka består av en frånterminal, en destinationsterminal och en lista med delsträckor som används. Den sparar även kostnaden, volymen och tiden. Dessa kan dock härledas från delsträckorna men sparas även i sträckan för snabbheten och enkelhetens skull.

En transporttyp består av en frånterminal, en destinationsterminal och en mängd med sträckor. Den har även tiden det tar att köra alla sträckor samt den maximala volymen som kan transporteras på den. Den maximala volymen är den delsträckan som har minst volym. Alla sträckor som ingår i en transporttyp körs på samma fysiska transport. Kostnaden för transporttypen blir då kostnaden för den längsta sträckan då det är den enda fysiska transport som att kommer att köras.

Figur 23 beskriver hur delsträckor, sträckor och en transporttyp är relaterade. En transporttyp består av en eller flera sträckor. Varje transporttyp har en sträcka som har samma start och destinationsterminal som transporttypen. Alla kanter mellan terminaler är delsträckor. De kanter och terminaler som är rödmarkerade är sträckor som ingår i transporttypen och den ljusblå är transporttypen. I exemplet i figuren betyder det att post som skall till från Alvesta till Västerås, från Alvesta till Karlstad och från Tomteboda till Västerås kan köras på samma fysiska transport.




Figur 23: Hur delsträckor, sträckor och transporttyper är relaterande.

Det finns även en lista med alla tillgängliga transporttyper. Denna lista är statisk och skapas när modellen skapas.

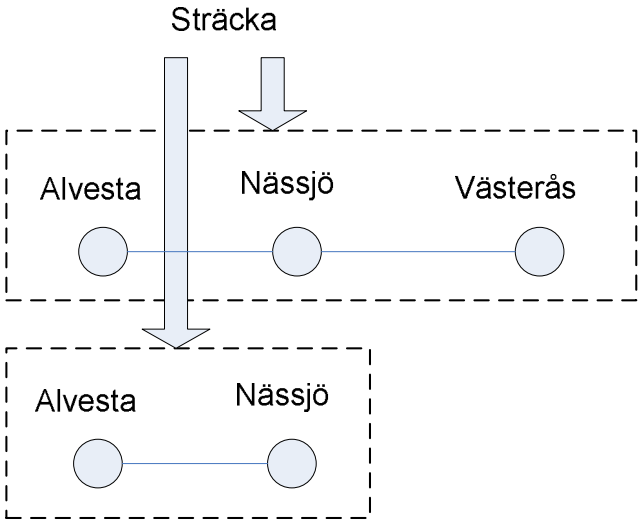
En transport består av en referens till en transporttyp och en starttid. Den är också den minsta beståndsdelen som kan specificeras i en kandidatlösning. Det går inte att köra en enskild delsträcka eller sträcka. Det går heller inte att specificera en transporttyp i en kandidatlösning då den inte har någon starttid.

En kandidatlösning är en lista med transporter, vilket motsvarar ett transportupplägg i PRiTSi. De används av optimeringsalgoritmen för att representera en potentiell lösning till problemet. Det är kandidatlösningen som modifieras för att lösa problemet.

Samlastning är en viktig del av simuleringen för utan den kommer orimligt dyra och långa kandidatlösningar skapas. Detta eftersom varje fysisk transport då enbart kan ha med post från en terminal till en annan terminal, samtidigt som dessa transporter kommer att vara dåligt utnyttjade med hänsyn till deras volymkapacitet. Därför är samlastning möjlig i den förenklade modellen.

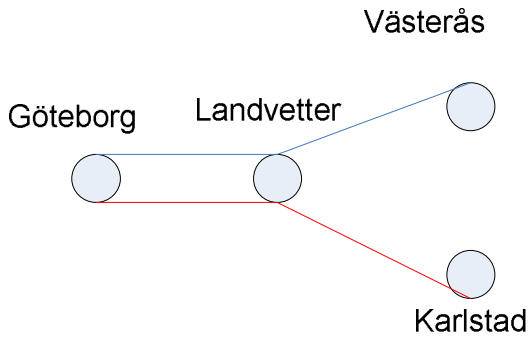
Samlastning åstadkoms genom transporttyper. Sträckorna i en transporttyp bestämmer vilka terminaler som post kan hämtas ifrån och lämnas av på. Alla sträckor i en transporttyp, utom den längsta, körs kostnadsfritt. Det är enbart den längsta sträckan som kostar då det är den enda fysiska transport som kommer att köras, posten som körs på de andra sträckorna går med den. För att två sträckor skall kunna samlastas krävs att en av sträckorna är en del av den andra. I exemplet i figur 24 är det två

sträckor, en från Alvesta till Västerås och en från Alvesta till Nässjö. Dessa kan samlastas då sträckan från Alvesta till Nässjö är en del av sträckan från Alvesta till Västerås.




Figur 24: Två stycken sträckor som kan samlastas

Omlastningspunkter kategori 1 representeras i den förenklade modellen som hubbar. Hubbar fungerar som Omlastningspunkter kategori 1 fungerar i PRiTSi, d.v.s. en transport kan aldrig ha sin slutdestination i en hubb. En sträcka kan inte specificeras att sluta i en hubb, men den kan dock ha en hubb som sista nod. Om sträckan slutar i en hubb kommer det att krävas en ny transport för att ta posten till sin slutdestination. Detta fungerar på detta sätt för att kunna samlasta transporter som har slutmål som skiljer sig åt, men som använder delvis samma terminaler och hubbar. Figur 25 beskriver ett exempel på detta. Detta exempel hade inte kunnat samlastas utan hubbar, då en av sträckorna inte fullt kan ersätta den andra.




Figur 25: Två sträckor som egentligen har olika slutdestinationer men som slutar i Landvetter för att kunna samlastas

En transporttyp som går från Göteborg till Västerås via Landvetter har två sträckor. Den första är från Göteborg till Västerås, men den slutar i en hubb (i detta fall Landvetter). Den andra sträckan går från Göteborg till Karlstad och den slutar i samma hubb (Landvetter). Dessa två sträckor kan samlastas i och med att de slutar i en hubb. Detta innebär att de får en gemensam delsträcka (en minsta gemensam nämnare) från Göteborg till och med Landvetter. Den egentliga slutdestinationen bortses ifrån och detta medför att det kommer det att krävas två nya transporter, en

från Landvetter till Västerås och en från Landvetter till Karlstad, för att få fram posten.

## 6.4 Post

I den förenklade modellen finns bara en postsort. Den största anledningen till detta är att hålla den förenklade modellen enkel. Ett exempel på när flera postsorter komplicerar saker är samlastning. Säg att det finns tre postsorter som i PRiTSi. Hur skall en transport lastas så den får med sig så mycket post som möjligt? Får att svara på den frågan måste det gå att värdera postsorterna mot varandra, en sådan värdering är inte alltid är självklar. Om värdeförhållandena skiljer sig från volymförhållandena liknar problemet som uppstår bin packing. Då måste ett bin packing problem lösas varje gång en transport skall lastas. Genom att bara ha en postsort undviks sådan problematik. För att omvandla postsorterna i PRiTSi till en postsort används omvandlingsfaktorer. Dessa omvandlingsfaktorer hämtas ifrån PRiTSi och är baseras på hur många av varje postsort som får plats i en postbehållare. En klump motsvarar fem stycken C4 brev och 30 stycken C5 brev.

Denna förenkling innebär att om postdistributionsnätverket har flera postsorter kan en lösning som löser problemet bra på den förenklade modellen fungerar dåligt på postdistributionsnätverket. Men om postsorterna enbart tar upp olika mycket plats, så behöver inte denna förenkling bli så betydelsefull.

## 6.5 Grovsimulering

För att kunna utvärdera en kandidatlösning krävs att den kan simuleras i modellen. Detta är uppgiften för grovsimuleringen.

Indatan till grovsimuleringen är en brevmatris och en kandidatlösning. Utdatan är kostnaden för transporterna och hur många brev som inte kommer fram till sin slutdestination. Den informationen används sedan av optimeringsalgoritmen för att fastställa hur bra kandidatlösningen är. Det sker ingen tolkning av resultatet i grovsimuleringen.

En händelse representerar en aktivitet som sker i grovsimuleringen. Grovsimuleringen utför händelser i den ordningen de inträffar och på så sätt simuleras händelseförloppet. En händelse består av en tid som talar om när den inträffar, en typ som säger vilken händelse det är, en fråntermінal samt en destinationsterminal. Den består också av ett tal som talar om hur mycket post som transporterades, detta tal bestäms först när händelsen utförs.

Det finns två typer av händelser, de är ”lasta in” och ”lasta av”. ”Lasta in” lastar in all post som finns på fråntermінalen eller om inte all post får plats så mycket som transporten kan lasta. ”Lasta av” lastar av all post som skall till destinationsterminalen.

Det tas ingen hänsyn till lasttider i simuleringen, detta innebär att en optimal lösning antagligen inte skulle vara optimal i PRiTSi då det tar en viss tid att lasta av och lasta in post. En transport som inkommer till en terminal med post och en transport som avgår direkt efter från den terminalen med posten kommer att fungera i modellen men inte i verkligheten. Det tas heller ingen hänsyn till den interna sorteringen på terminaler. Ibland måste post hanteras och sorteras innan den kan forslas vidare. Detta kommer inte den förenklade modellen att simulera.

Algoritmen som sköter grovsimuleringen använder sig av ”discrete event simulation”. För en introduktion till ”discrete event simulation” se (Ball, 2001). Innan någon simulering sker skapas först alla händelser. För att skapa händelserna går den igenom alla sträckor i alla transporter och varje sträcka blir två händelser, en ”lasta in” och en ”lasta av”. Starttiderna för sträckorna räknas ut ifrån när transporten startar plus eventuella delsträckor som är före den aktuella sträckan. Detta är samma tid som händelsen ”lasta in” händer. Händelsen ”lasta ut” räknas ut genom att ta starttiden för transporten och addera den med tiden det tar att åka sträckan. Hur mycket post som tas med bestäms först när händelsen utförs.

Denna simuleringsmetod valdes för den på ett enkelt sätt serialiserar en process som sker parallellt.

Ett alternativ hade varit att sträckorna i alla transporterna sorteras efter starttid och läggs i en lista och utförs en efter en, utan att skapa några händelser. Problemet med det är att det blir svårt att hålla reda på hur mycket post sträckorna har tagit upp, då samma sträcka kan utföras av olika transporter parallellt.

Ett annat alternativ till ”discrete event simulation” är att simulera tiden. Simuleringen startar då vid en given tidpunkt och de transporter som har den tiden som starttid lastar den post de kan ta med sig. Sen flyttas klockan fram ett givet intervall och alla transporter som skall starta vid detta klockslag startar och om någon transport kommit fram lastar den av sin post. Simuleringen fortsätter sedan på detta sätt tills alla transporter kommit fram. Nackdelen med denna metod är att veta hur mycket tiden skall flyttas fram varje steg så att inga transporter missas och så inte simuleringen tar orimligt lång tid.

För att simulera samlastning tilldelas varje transport ett gemensamt utrymme. Vid händelsen ”lasta in” läggs post till i det gemensamma utrymmet och vid ”lasta av” tas post bort från det.

Ett scenario har en deadline som måste hållas. Alla händelser som startar efter deadline utförs inte.

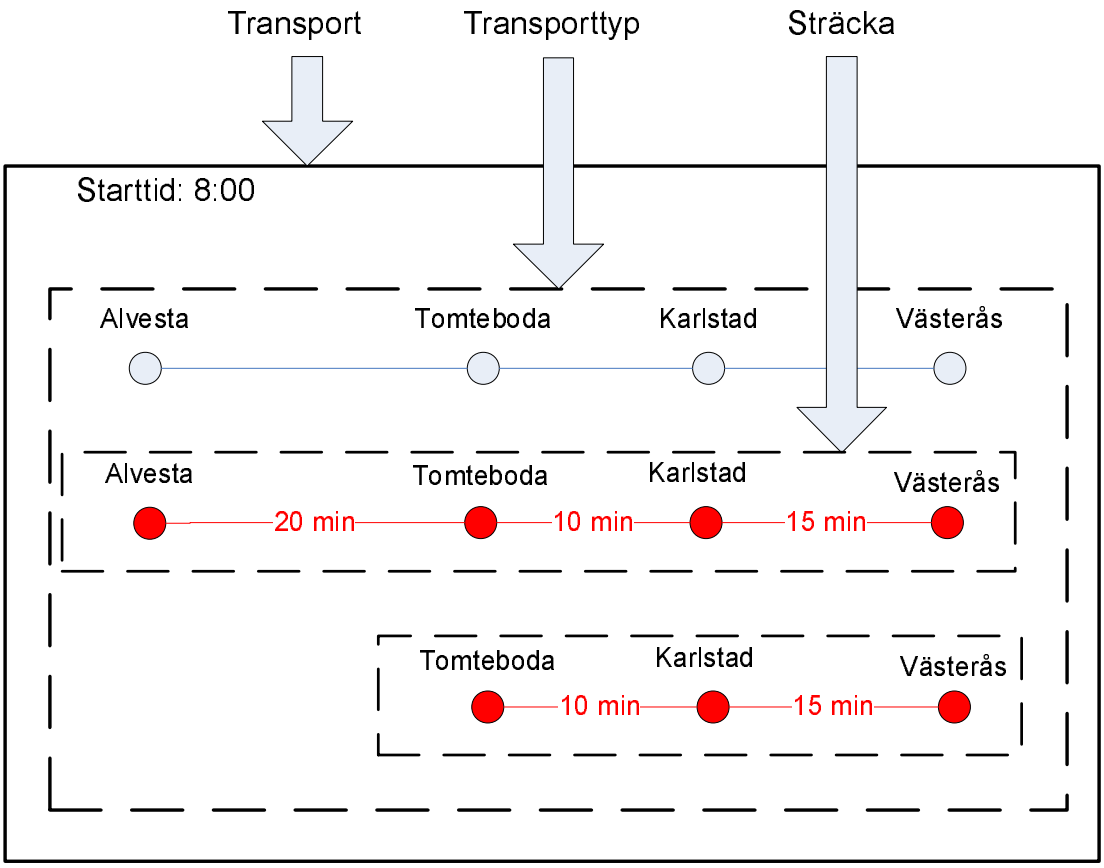
### 6.5.1 Grovsimuleringen i pseudokod

```
Skapa en lista med transportttyper.
Sätt den totala kostnaden till noll.
För varje transport i kandidatlösningen
      Öka den totala kostnaden med kostnaden för transporten.
      Associera ett transportutrymme med transporten.
      För varje sträcka i transporten
            Räkna ut starttiden för den sträckan.
            Skapa en referens till transportutrymmet.
            Lägg till transporten i transporttypslistan.
      slut
slut
Skapa en tom lista med händelser.
För varje sträcka i transporttypslistan
      Skapa en händelse ”lasta in”, fråntерminalen är sträckans
      fråntерminal, tillterminalen är sträckans tillterminal och
      tiden för händelsen är starttiden för sträckan.
```

```
      Skapa en händelse ”lasta ut”, fränterminalen är sträckan
      fränterminal, tillterminalen är sträckans tillterminal och
      tiden för händelsen är starttiden för sträckan plus tiden det
      tar för sträckan att köras.
slut
För varje händelse i listan med händelser
      Om händelsen är ”lasta in”
            Hämta antalet brev som skall från händelsens fränterminal
            till händelsens tillterminal från brevmatrisen.
            Hämta när breven finns tillgängliga.
            Om händelsen inträffar före deadline och breven finns
            tillgängliga innan händelsen inträffar
                  Sätt kapaciteten till den maximala volymen som
                  sträckan kan lasta minus antalet brev i sträckans
                  gemensamma utrymme.
                  Om kapaciteten är mindre än antalet brev
                        Sänk antalet brev i brevmatrisen med
                        kapaciteten.
                        Öka det gemensamma utrymmet med kapaciteten.
                        Sätt antalet brev som transporteras till
                        kapaciteten.
                  Annars
                        Öka på det gemensamma utrymmet med antalet
                        brev i brevmatrisen.
                        Sätt antalet brev i brevmatrisen till noll.
                        Sätt antalet brev som transporteras till
                        antalet brev som togs med.
                  slut
      Om händelsen är ”lasta ut”
            Om till terminalen är en hubb
                  Minska det gemensamma utrymmet med antalet
                  transporterade brev.
                  Lägg till breven i brevmatrisen. Där fränterminalen
                  är hubben och tillterminalen är postens
                  slutdestination.
            Annars
                  Minska det gemensamma utrymmet med antalet
                  transporterade brev.
            slut
      slut
slut
```

I exemplet i figur 26 så startar transporten klockan 8:00 och det tar 20 minuter att åka från Alvesta till Tomteboda, så sträckan från Tomteboda till Västerås startar klockan 8:20. Transporten kommer då att vara framme i Västerås 8:45. Händelsen ”lasta in” Tomteboda inträffar då 8:20 och händelsen ”lasta ut” Västerås 8:45.

Den andra sträckan från Alvesta till Västerås startar klockan 8:00 och kommer fram 8:45. ”Lasta in” händelsen sker då 8:00 och ”lasta ut” 8:45.




Figur 26: Ett transportexempel.

När alla händelser har skapats läggs de i en lista (figur 27) och sorteras efter vilken tid de inträffar, i stigande ordning. Denna lista itereras sedan igenom och händelserna utförs i den ordningen de påträffas. T.ex. om den första händelsen är samma händelse som i exemplet ovan kommer grovsimuleringen först att kolla på vad det är för typ av händelse. Det är en ”lasta in”-händelse så den kollar om den kan lasta in all post från Tomteboda till Västerås på transporten, om det lyckas flyttas all post till transporten. Om transporten inte kan ta all post tar den med så mycket den har plats med. Sen går simuleringen vidare till nästa händelse och kollar vad den har för typ. Det är en ”lasta av”-händelse så den tar det antalet post som den tog upp i Tomteboda och släpper av det i Västerås. I simuleringen läggs inte breven tillbaka i brevmatrisen då detta skulle betyda att breven ska tillbaka till Tomteboda igen. Men om den sista sträckan slutar i en hubb läggs breven till i brevmatrisen, för transporten är slut men posten är inte framme vid sin slutdestination. Sedan tar simuleringen nästa händelse och utför den och detta upprepas tills det inte finns några mer händelser i listan. Simuleringen är då klar.

| Tid | Fråntermina l | Destinationsterminal | Händelse typ |
|---|---|---|---|
| 08:00 | Alvesta | Västerås | Lasta in |
| 08:20 | Tomteboda | Västerås | Lasta in |
| 08:45 | Alvesta | Västerås | Lasta ut |
| 08:45 | Tomteboda | Västerås | Lasta ut |

Figur 27: Den sorterade händelselistan från figur 20:

Händelselistan bör sorteras med en stabil sorteringsalgoritm, d.v.s. en algoritm som bevarar ordningen på element som är likadana, för annars kan tillförandet eller borttagandet av en transport påverka resultatet av händelser som är oberoende av den transporten. Som ett exempel, antag att det finns 10 000 brev som skall från Nässjö till Umeå, 5 000 brev som skall från Nässjö till Karlstad och 50 000 brev som skall från Nässjö till Årsta. Låt det finnas en transporttyp som består av sträckorna från Nässjö till Umeå, från Nässjö till Karlstad och från Nässjö till Årsta. Antag också att den maximala volymen som kan transporteras på den transporttypen är 25 000. Denna transporttyp kommer att bli tre stycken lasta in händelser och de kommer alla att inträffa vid samma tidpunkt. Det kommer givetvis att bli tre stycken ”lasta ut” händelser också, men dessa bortses ifrån då de inte är relevanta för detta exempel. Låt det finnas ytterligare en transporttyp som går enbart från Nässjö till Årsta och som har en starttid som är senare än den första transporttypen. Låt den maximala volymen på den transporttypen vara 50 000. Det finns nu fyra stycken lasta in händelser, av dem har tre samma tid. Antag nu att händelserna sorteras i den ordning som visas i figur 28:

| Transport | Händelse | Från | Till | Tid | Terminal | Transport | Utrymme |
|---|---|---|---|---|---|---|---|
| 1 | Lasta in | Nässjö | Årsta | t | 50 000 | 25 000 | 0 |
| 1 | Lasta in | Nässjö | Umeå | t | 10 000 | 0 | 0 |
| 1 | Lasta in | Nässjö | Karlstad | t | 5 000 | 0 | 0 |
| 2 | Lasta in | Nässjö | Årsta | t+1 | 25 000 | 25 000 | 25 000 |

Figur 28: Exempel på en utförd händelselista

Först kommer händelsen ”lasta in” från Nässjö till Årsta vid tid *t* att inträffa. Det finns 50 000 brev som skall från Nässjö till Årsta och den maximala volymen för transporttypen är 25 000 så endast 25 000 brev tas med. Därefter sker ”lasta in” från Nässjö till Umeå vid tid *t*. Det finns 10 000 som skall från Nässjö till Umeå men transporttypen är redan full så inga brev tas med. Sen sker ”lasta in” från Nässjö till Karlstad vid tid *t*. Det finns 5 000 brev som skall från Nässjö till Karlstad men transporttypen är redan full så inga brev tas med. Sist sker ”lasta in” från Nässjö till Årsta vid tid $t+1$. Det finns 25 000 brev som skall från Nässjö till Årsta och den maximala volymen för transporten är 50 000 så all post tas med. I detta händelseförlopp blir det 5 000 brev kvar i Nässjö som skulle till Karlstad och 10 000 brev som skulle från Nässjö till Umeå.

Antag nu att händelselistan istället sorteras i denna ordning, detta p.g.a. av att en oberoende transport har lagts till och på så vis fått den ostabila sorteringsalgoritmen att sortera händelserna i ordningen som visas i figur 29:

| Transport | Händelse | Från | Till | Tid | Terminal | Transport | Utrymme |
|---|---|---|---|---|---|---|---|
| 1 | Lasta in | Nässjö | Umeå | t | 10 000 | 10 000 | 15 000 |
| 1 | Lasta in | Nässjö | Karlstad | t | 5 000 | 5 000 | 10 000 |
| 1 | Lasta in | Nässjö | Årsta | t | 25 000 | 10 000 | 0 |
| 2 | Lasta in | Nässjö | Årsta | t+1 | 40 000 | 40 000 | 10 000 |

Figur 29: Exempel på en utförd händelselista

Detta kommer att innebära att först sker ”lasta in” från Nässjö till Umeå vid tid *t*. Det finns 10 000 brev som skall från Nässjö till Umeå och transporttypen kan maximalt ta

med 25 000 brev så alla brev får plats. Nästa händelse är lasta in från Nässjö till Karlstad vid tid *t*. Det finns 5 000 brev som skall från Nässjö till Karlstad och det finns 10 000 brev i transporttypen så alla brev får plats. Nästa händelse är ”lasta in” från Nässjö till Årsta vid *t*. Det finns 50 000 brev som skall från Nässjö till Årsta och det finns 15 000 brev i transporttypen och den maximala volymen för transporttypen är 25 000 så bara 10 000 brev får plats. Därefter sker lasta in från Nässjö till Årsta vid tid $t+1$. Det finns 40 000 brev kvar som skall från Nässjö till Årsta och transporttypen tar 50 000 så alla brev kan tas med. I detta händelseförlopp transporteras alla brev.

Den oberoende transporten har alltså fått händelserna att inträffa i en annan ordning. Den ordningen är inte bara annorlunda utan också bättre. Även om den oberoende transporten inte transporterar någon post själv så bidrar den till att kandidatlösningen blir bättre. Detta kan utnyttjas av optimeringsalgoritmen genom att lägga till transporter enbart därför att de ändrar ordningen på händelser vilket kommer att ge lösningar med överflödiga transporter.

Det kan alltså bli två stycken helt olika händelseförlopp beroende på hur händelselistan sorteras, därför bör en stabil sorteringsalgoritm användas.

Kostnaden för en kandidatlösning är oberoende av hur simuleringen gick, d.v.s. transporterna kostar lika mycket oavsett om de verkligen tog med någon post eller inte. För att räkna ut kostnaden summerar grovsimuleringen kostnaden för alla transporttyper i alla transporter.

Mängden post som inte kom fram till sin destinationsterminal räknas fram genom att iterera igenom brevmatrisen och summera alla brev som är kvar.

# 7 Optimering

## 7.1 Val av algoritm

Genetiska algoritmer är enkla att använda, kräver inte så mycket resurser och är bra på att hitta lösningar på svåra problem (Gen och Runewi, 2000). De har även applicerats på det liknande problemet VRP med framgång (Tavares et al., 2001). Nackdelen med genetiska algoritmer är att det finns många parametrar som det är svårt att hitta lämpliga värden på. Hur dessa parametrar sätts kan ha stor inverkan på hur effektiv en genetisk algoritm är. De söker i hela sökrymden utan någon information om problemet vilket gör att de kan ta längre tid på sig att hitta bra lösningar gentemot algoritmer som utnyttjar information om problemet. En bra lösning är en lösning som transporterar alla brev och är relativt billig

Mycket av styrkan hos genetiska algoritmer ligger i en effektiv crossover-operator (Whitley, 1994). Den representation som valts har dock ingen självklar crossover-operator. Om man använder någon av de vanliga crossover-operatorerna, som t.ex. one-point crossover, med den befintliga representationen är det osannolikt att två lösningar som är halvbra kan kombineras till en bra lösning. Detta eftersom det inte finns någon relation mellan vart de finns i listan och vad det är transporten gör. Det som eftersträvas med en crossover är att ta en lösning som löser en del av problemet bra och kombinera den med en annan lösning som löser en annan del av problemet bra. För att sedan kombinera dessa två för att få en lösning som löser bägge problemen bra. Problemet med den representation som valts är det inte lätt går att dela upp problemet i delar. Det är svårt att veta exakt vilken del av problemet som en eller flera transporter löser. Därför är det svårt att kombinera ihop två halvbra lösningar till en bra lösning.

Eftersom en genetisk algoritm utan en bra crossover-operator förlorar mycket av sin styrka kommer hill climbingalgoritmen att användas för att den är enkel och lätt att implementera. Trots att den i teorin bara kan hitta lokala optimum tros den kunna prestera bra på detta problem om heuristiken är bra, speciellt om startlösningen är tillräckligt bra. En annan anledning är att det inte finns så många parametrar i hill climbingalgoritmen, detta medför att det blir lättare att skilja på som är resultatet av heuristiken och vad som är resultatet av algoritmen.

## 7.2 Hill climbing

Indatan till hill climbingalgoritmen är en kandidatlösning och antal iterationer den skall köras. Utdatan är en kandidatlösning. Heuristiken som undersöks kommer att appliceras på mutationsoperatorn, därför kommer tre stycken olika mutationsoperatorer skapas för att kunna utvärdera heuristiken. De mutationsoperatorer som skapas är slumpmässig, heuristisk och mixad.

Nedan följer hill climbingalgoritmen i pseudokod. Algoritmen körs ett bestämt antal varv eller en viss tid. Varje varv muteras den bästa lösningen minst en gång. Sannolikheten att det sker två mutationer är $0.5$, sannolikheten att det sker tre mutationer är $0.5^2$ och sannolikheten att det sker *n* mutationer är $0.5^{n-1}$. Anledningen till att mutationer kan ske flera gånger är att minska risken att hamna i lokala optima. Om bara en mutation kan ske varje iteration kommer det t.ex. innebära att hubbarna inte kommer att användas. Detta eftersom posten måste färdas i två steg för att komma fram. En mutation kommer att kunna flytta posten första steget men

målfunktionen kan inte se att den transporten gjorde någon nytta. För att undvika detta problem kan mutationer ske flera gånger per iteration.

```
Sätt den bästa lösningen till den givna lösningen.
Kör den bästa lösningen med grovsimuleringen.
Evaluera resultatet från grovsimuleringen med målfunktionen.
Spara undan resultatet som det bästa resultatet.
Sätt antalet iterationer till noll.
Så länge som antalet iterationer är mindre än det maximala antalet
iterationer eller den maximala tiden har passerat
   Gör en kopia på den bästa lösningen.
   Gör
         Mutera kopian av den bästa lösningen
   Så länge slumpmässigt tal modulus två är lika med noll
   Kör den muterade lösningen med grovsimuleringen.
   Evaluera den muterade lösningen med målfunktionen.
   Om resultatet från den muterade lösningen är bättre än den
   dittills bästa lösningen
         Den bästa lösningen sätts till den muterade lösningen.
         Den muterade lösningens resultat sparas undan som det
         bästa resultatet.
   slut
   Öka på antalet iterationer med ett.
slut
Returnera den bästa lösningen.
```

## 7.3 Slumpmässig mutationsoperatorn

I den slumpmässiga mutationsoperatorn kan fyra stycken olika mutationer ske. De är:

- Ändra vilken transporttyp som en transport använder
- Ändra tiden då en transport avgår
- Ta bort en transport
- Lägg till en transport

Varje gång mutationsoperatorn körs utförs exakt en av dessa mutationer. Vilken av operationerna det blir bestäms av slumpen. De tre första mutationerna påverkar exakt en transport, vilken transport det blir slumpas fram. ”Lägg till transport” mutationen är lite annorlunda då den inte förändrar en befintlig transport, utan den lägger alltid till exakt en transport.

Vid mutation av transporttypen slumpas en ny transporttyp fram ur listan med alla transporttyper. När tiden skall muteras slumpas en ny starttid för transporten fram. Vid borttagning tas en slumpvis transport bort. Om en ny transport skall läggas till slumpas först vilken transporttyp den skall använda sig av och sedan vilken starttid den skall ha.

## 7.4 Heuristisk mutationsoperator

Idén med den heuristiska mutationsoperatorn är att, istället för att göra slumpmässiga val vid muteringen, basera valen på resultatet från den förra iterationen.

Samma muteringstyper används i den heuristiska mutationen som vid den slumpmässiga, d.v.s. ändra vilken transporttyp en transport använder, ändra avgångstiden för en transport, ta bort en transport och lägg till en transport.

Grovsimuleringen returnerar två listor, utöver den vanliga utdatan, dessa används för att göra mutationsvalen. Den ena listan innehåller alla transporter som inte transporterade någon post (tomlistan) och den andra innehåller information om vart det ligger kvar post (postkvarlistan).

Vid mutation av avgångstiden för en transport slumpas en transport ur tomlistan fram. Denna transport får en ny avgångstid som också slumpas fram. Tanken med denna mutation är att en transport som inte gjorde något förra iterationen kanske kördes vid fel tidpunkt, t.ex. före det fanns någon post vid den terminalen.

När mutation av vilken transporttyp en transport skall använda sker, slumpas först en transport fram ur tomlistan. Den nya transporttypen för den transporten slumpas sedan fram. Tanken med denna mutation är att en transport som inte gjorde något förra iterationen kanske använde fel transporttyp, d.v.s. den körde post mellan två terminaler som inte behövde det.

Vid borttagande av en transport slumpas först den transport som skall påverkas fram ur tomlistan. Den transporten tas sedan bort. Tanken med denna mutation är att en transport som inte gjorde något förra iterationen kanske är överflödig.

När en transport skall läggas till slumpas först ett element ur postkvarlistan fram. Det elementet innehåller vilken terminal som posten finns på och destinationsterminalen. Sedan sker en sökning i listan med alla transporttyper. Alla transporttyper med matchande frånterminal och destinationsterminal läggs i en temporär lista. En transporttyp slumpas sedan fram ur den temporära listan. Därefter hämtas tiden då posten finns tillgänglig och en avgångstid slumpas fram som är senare än när posten finns tillgänglig och tidigare än deadline. Transporttypen och avgångstiden läggs sedan till som en ny transport. Denna mutation medför att en transport har lagts till som bör minska antalet brev som blir kvar vid nästa simulering.

Om tomlistan är tom sker det alltid en tilläggning av en transport men en tilläggning av en transport sker endast om postkvarlistan inte är tom. Detta innebär att om kandidatlösningen transporterar alla brev och inga transporter är överflödiga kommer den heuristiska mutationsoperatorn inte att göra någonting.

## 7.5 Mixad mutationsoperator

Den slumpmässiga mutationsoperatorn ändrar helt slumpmässigt i kandidatlösningen och eftersom det är en stor sökrymd kan den ha svårt att hitta lösningar som transporterar all post. Den har dock inte några förutfattade meningar om problemet och kan hitta dellösningar som är bra, men inte uppenbara.

Den heuristiska mutationsoperatorn ändrar däremot bara när den tror att den kan förbättra kandidatlösningen. Detta innebär att den snabbt kommer att hitta en lösning som transporterar alla brev och inte använder några transporter som inte kör någon post. Den kommer däremot inte att försöka förbättra en lösning efter att den hittat en som får fram all post.

Den mixade mutationsoperatorn använder både den slumpmässiga och den heuristiska mutationsoperatorn för att snabbt hitta en bra lösning och sedan kunna förbättra den. Det är 50 % chans att den ena eller andra mutationsoperatorn används. Meningen med den mixade mutationsoperatorn är den skall kombinera det bästa från de två andra mutationsoperatorerna. Det är den mutationsoperator som har bäst förutsättningar att lyckas bra. Detta för att heuristiken kommer att styra lösningen snabbare mot en lösning som transporterar alla brev, medan den slumpmässiga kommer att föreslå ändringar som är bra men inte uppenbara.

## 7.6 Målfunktionen

Målfunktionen räknar ut målvärdet utifrån antalet brev som är kvar och kostnaden för lösningen. Den använder vikter som den multiplicerar med antalet brev som är kvar och kostnaden. Hur dessa vikter sätts har stor inverkan på hur optimeringen sker. Värdet från målfunktionen skall minimeras i detta problem.

Vikten som multipliceras med antalet brev som inte kommer fram sätts högt i förhållande till vikten som multipliceras med kostnaden, eftersom det är allra viktigast för en lösning att alla brev kommer fram. Förhållandet som används är 1:1 000. Varje brev som inte kommer fram kostar alltså 1 000 kr.

# 8 Optimeringsresultat

Två stycken olika experiment utförs, en med ett fast antal iterationer och en med en bestämd tid. Det första experimentet är till för att se hur de olika mutationsoperatorerna presterar på samma antal iterationer. Det andra experimentet utförs för att ta reda på hur bra lösningar de olika mutationsoperatorerna kan hitta på samma tid. Detta är intressant för den heuristiska mutationsoperatorn utför mer beräkningar varje iteration än den slumpmässiga. Detta experiment kan ta reda på vilken inverkan detta har på lösningarna som hittas inom den bestämda tiden. Experimenten utförs på en vanlig PC med operativsystemet Windows XP Professional Service Pack 2. Andra processer kördes samtidigt som experimenten men deras inverkan antas vara försumbar.

I de två experimenten utförs tre olika delexperiment, ett för varje mutationsoperator.

## 8.1 Hill climbing med fast antal iterationer

Varje delexperiment körs tio omgångar. Det startar från en tom lösning och körs i 40 000 iterationer. Från de tio körningar räknas ett medelvärde av resultaten fram.

Resultaten från körningarna är ”Kostnad”, ”Antal brev kvar” och målvärde.

”Kostnad” är hur mycket en lösning kostar. ”Antal brev kvar” är hur många brev som inte transporterades till sin slutdestination. Det inkluderar brev som ligger kvar på hubbar. Målvärdet är värdet från målfunktionen i hill climbingalgoritmen. Detta värde skall vara så lågt som möjligt.

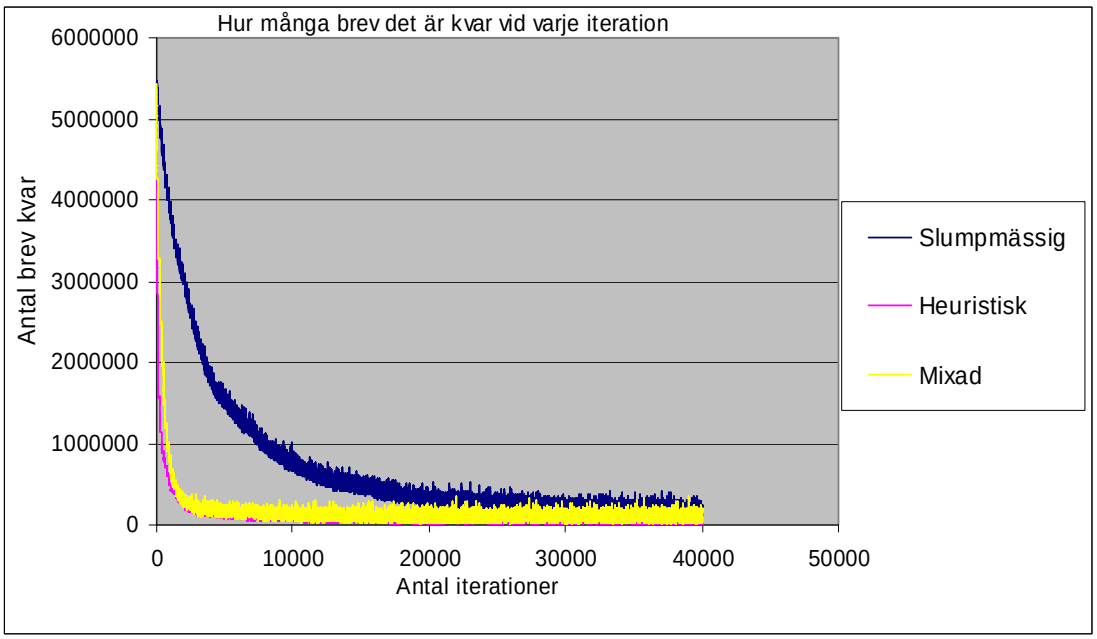
Den slumpmässiga mutationsoperatorn hittade ingen lösning som transporterade all post i någon av tio omgångarna. Den heuristisk hittade fyra lösningar av tio medan den mixade hittade åtta av tio.

I figur 30 visas medelvärdena av de bästa resultaten från respektive körning.

| Mutationsoperator | Kostnad | Antal brev kvar | Målvärde |
|---|---|---|---|
| Slumpmässig | 1 536 042 | 56 543 | 5808 |
| Heuristisk | 3 295 656 | 20 458 | 638 |
| Mixad | 1 402 657 | 2 900 | 445 |

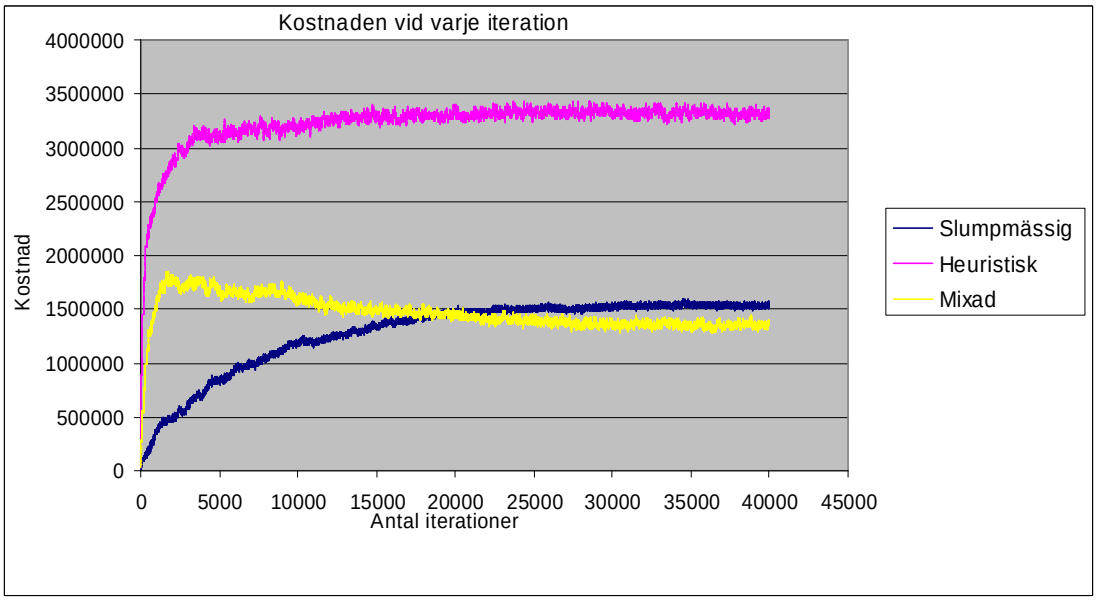
Figur 30: Medelvärdena av resultaten för respektive mutationsoperator

Figur 31 visar antal brev det är kvar vid varje iteration för respektive mutationsoperator. Det är medelvärdet av varje iterations ”antal brev kvar”, precis efter mutationen, som visas. Det är därför som den varierar både uppåt och nedåt.



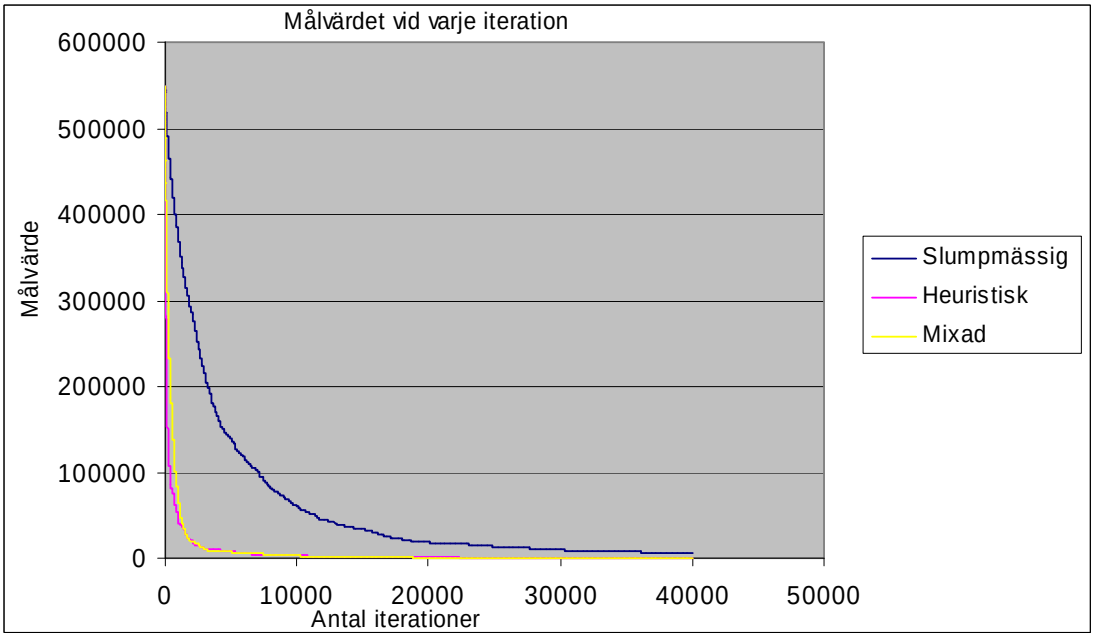

Figur 31: Antal brev det är kvar vid varje iteration

Figur 32 visar kostnaden vid varje iteration. Det är medelvärdet av varje iterations kostnad, precis efter mutationen, som visas. Det är därför som den varierar både uppåt och nedåt.




Figur 32: Kostnaden vid varje iteration.

Figur 33 visar målvärdet vid varje iteration.




Figur 33: Målvärdet vid varje iteration.

## 8.2 Hill climbing i en bestämd tid

Varje delexperiment körs tio omgångar i vardera 15 minuter och medelvärdena av de bästa lösningarna presenteras nedan.

I figur 34 visas medelvärdena av de bästa lösningarna från respektive mutationsoperator. Målvärdet är värdet från målfunktionen, det skall vara så lågt som möjligt. Iterationer är antalet iterationer som har körts.

| Mutationsoperator | Målvärde | Iterationer |
|---|---|---|
| Slumpmässig | 1 116 | 26 328 |
| Heuristisk | 683 | 135 437 |
| Mixad | 137 | 59 588 |

Figur 34: Medelvärdena av de bästa lösningarna.

# 9 Slutsats och diskussion

Den mesta av tiden gick åt till att ta fram och testa den förenklade modellen. Resultatet blev en enkel och snabb modell, vilka var de största anledningarna till att modellen skapades.

Hur väl den förenklade modellen överensstämmer med simuleringsmodellen som den utgick ifrån (PRiTSi) är svårt att säga då för lite tester har utförts. De få testar som gjorts visar att kostnaden stämmer bra överens, men antalet brev kvar stämmer inte lika bra. Detta kan bero på omvandligen till bara en postsort i den förenklade modellen.

Om den förenklade modellen kan appliceras på andra postdistributionsnätverk än PRiTSi är också svårt att avgöra. Det krävs antagligen förändringar i modellen, då alla postdistributionsnätverk är lite annorlunda. Problemet är att en alltför generell modell inte klarar av att utnyttja de olika specialiteterna i verkliga postdistributionsnätverk, som behövs för att hitta en lösning som fungerar för dessa och inte bara på den förenklade modellen. En alltför specifik modell kan dock inte enkelt anpassas för olika postdistributionsnätverk.

Den allra viktigaste för en lösning är att få fram alla brev. Hill climbingalgoritmen med den slumpmässiga mutationsoperatorn applicerad på en tom lösning misslyckades i alla tio körningar att hitta en lösning som transporterar alla brev. Den heuristiska mutationsoperatorn hittade fyra gånger av tio en lösning som tog med alla brev. Den mixade mutationsoperatorn lyckades åtta av tio gånger hitta en lösning som tog med alla brev.

Kostnaden är också viktigt för en lösning, när antalet brev som är kvar är noll skall optimering ske mot kostnaden. Detta syns tydligast i den mixade operatorn. Först går kostnaden upp kraftigt, i den fasen läggs transporter till. Sen när en lösning har hittats som transporterar all post börjar optimeringen mot kostnaden och kostnaden sjunker tydligt. Kostnaden blir till och med lägre än för den slumpmässiga mutationsoperatorn, fast den slumpmässiga operatorn inte tar med alla brev. Kostnaden för den heuristiska mutationsoperatorn ökar kraftigt i början då den bygger upp lösningen. När sedan de flesta brev transporteras planar den ut och den sjunker knappt aldrig, detta beror på att när den hittat en lösning som transporterar alla brev och inte använder några överflödiga transporter slutar den förändra lösningen. Den slumpmässiga operatorn lyckas aldrig att hitta en lösning som transporterar alla brev så den utför aldrig någon fokuserad optimering mot kostnaden.

Målvärdet för både den heuristiska och den mixade mutationsoperatorn närmar sig noll snabbare än den slumpmässiga mutationsoperatorn. Dessa resultat indikerar på att det går fortare att hitta en bra lösning med heuristik för detta problem. De visar också att den mixade mutationsoperatorn fungerar bäst. Detta tyder på att det är bra att styra algoritmen men samtidigt ha kvar ett element av slump. En anledning till att denna heuristik är bra för detta problem är att algoritmen fortare hittar en bra lösning, som den sedan kan optimera.

Slutsatsen som kan dras är att heuristiken sänker tiden det tar att hitta en bra lösning. Detta kan ses i det andra experimentet där varje mutationsoperator har körts lika länge. Den mixade presterade bäst, d.v.s. den hade lägst medelvärde på målvärdet, följt av heuristiken och den slumpmässiga sist. Den är bara testad på ett scenario men den har potential att fungera för andra scenarion också, då heuristiken och scenariot inte har några uppenbara samband.

Hur bra lösningarna fungerar på PRiTSi går inte att säga utan testning. Detta har inte gjorts på grund av problem med vissa simuleringar i PRiTSi.

Något att notera är den mixade och den heuristiska mutationsoperatorn hinner med många fler iterationer gentemot den slumpmässiga. Detta för att den heuristiska har mindre antal transporter i dess kandidatlösningar än den slumpmässiga, för den lägger inte till några transporter som inte transporterar någon post. Detta medför att grovsimuleringen kommer gå snabbare då det inte finns så många transporter att simulera. Vilket innebär att mer iterationer kan köras i den heuristiska mutationsoperatorn trots att den gör fler beräkningar än den slumpmässiga.

Det fanns lösningar som lämnade kvar brev som värderades bättre än lösningar som inte gjorde det. Det tyder på att vikterna behöver justeras.

## 9.1 Fortsatt arbete

Den förenklade modellen behöver i ett första steg testas mera mot PRiTSi för att validera den. Efter det skulle den behöva testas mot ett helt annat postdistribueringsnätverk för att ta reda på om den är generell. Den förenklade modellen kan utvecklas mer för att bland annat kunna hantera flera postsorter.

Något som behöver undersökas mer är om lösningarna som den slumpmässiga mutationsoperatorn hittar skiljer sig gentemot lösningarna som hittas av den mixade och den heuristiska mutationsoperatorn när de appliceras på PRiTSi.

Något som också är intressant att undersöka är om andra representationer och algoritmer kan användas för att få bättre resultat, t.ex. en genetisk algoritm. Skulle de resultaten i så fall vara bättre än de som fås av hill climbingalgoritmen.

En modifikation av den mixade mutationsoperatorn är att använda heuristiken ända tills en lösning som transporterar alla brev hittas och då byta till den slumpmässiga. En annan variant är att sannolikheten att använda heuristiken står i relation till hur många brev som det är kvar, om det t.ex. är många brev kvar så kan det vara större sannolikhet att använda heuristiken och om det är få eller inga brev kvar så är det större sannolikhet att använda den slumpmässiga.

Det går även att använda en mer avancerad viktfördelning i målfunktionen. Vikten för antalet brev kvar kan vara baserad på avståndet till dess destinationsterminal. En slags uppskattning hur stor ansträngning det krävs för att transportera breven. Tanken är då att om breven ligger nära sin destinationsterminal så kommer det inte att innebära lika stor bestraffning. Tiden kan användas på samma sätt, om tiden posten är framme är strax efter deadline så blir det mindre bestraffning till lösningen än om posten är framme långt efter deadline.

# 10 Referenser


Alba, E. och Dorronsoro, B., 2004. Solving the Vehicle Routing Problem by Using Cellular Genetic Algorithms. In *Combinatorial Optimization, 4th European Conference EvoCop 2004,* Coimbra, Portugal. Lecture Notes in Computer Science 3004, 11-20,

Ball, P. 2001. Introduction to Discrete Event Simulation. In *2nd DYCOMANS workshop on "Management and Control : Tools in Action",* Algarve, Portugal. 15th - 17th May 1996, pp. 367-376

Barnhart, C., Johnson, E. L, Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H. 1996. Branch-and-Price: Column Generation for Solving Huge Integer Programs. *Georgia Institute of Technology. School of Industrial and Systems Engineering.*

Callan, R. 1999. *The Essence of Neural Networks.* Prentice Hall, Europe.

Chan, F.T.S. och Chung, S.H., 2004. Multi-criteria genetic optimization for distribution network problems. *Int J Adv Manuf Technol,* (2004) 24, pp. 517–532

Crainic, T.G., 2002. A Survey of Optimization Models for Long-Haul Freight Transportation, *Département des sciences administratives Université du Québec à Montréal.*

Dantzig, G.B. och Ramser, R.H., 1959. The Truck Dispatching Problem. *Management Science,* 6, pp. 80–91

Donati, A.V., Gambardella, L.M., Rizzoli, A.E., Casagrande, N. och Montemanni, R., 2002. Time Dependent Vehicle Routing Problem with an Ant Colony System. *Istituto Dalle Molle di Studi sull'Intelligenza Artficiale*

Ekberg, J. och Stablum, P. 2006. PRiTSi. Logistik, Samordning & Utveckling. Posten Sverige. (Power Point Presentation)

Franklin, J. 1983. Mathematical Methods of Economics. *The American Mathematical Monthly,* 90(4), pp. 229–244.

Gen, M och Runewi, C. 2000. *Genetic Algorithms & Engineering Optimization.* Wiley-Interscience, USA, Canada.

Georgakopoulos, A. och Mihiotis, A., 2004. Distribution networkdesign : an integer programming approach. *Journal of Retailing and Consumer Services* 11 (2004), pp. 41–49

Grunert, T. och Sebastian, H.J., 2000. Planning models for long-haul operations of postal and express shipment companies. *European Journal of Operational Research* 122 (2000) pp. 289-309

Haghania, A. och Jung, S., 2005. A dynamic vehicle routing problem with time-dependent travel times. *Computers & Operations Research,* 32, pp. 2959–2986

Hollis, B.L., Forbes, M.A. och Douglas, B.E. 2005. Vehicle routing and crew scheduling for metropolitan mail distribution at Australia Post. *European Journal of Operational Research*

Jablonsky, J. och Lauber, J. 1999. A Time–Cost Optimization of the National Postal Distribution Network. *Journal of multi-criteria decision analysis.* 8 (1999), pp. 51–56

Kallehauge, B. och Larsen, J., Madsen, B.G., 2005. Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations research*, 33(2006), pp. 1464–1487

Machado, P., Tavares, J., Pereira, F.B. och Costa, E. 2001. Vehicle Routing Problem: Doing it the Evolutionary Way. *Instituto Superior de Engenharia de Coimbra. Portugal.*

Mitchell, J.E., 1999. Branch-and-Cut Algorithms for Combinatorial Optimization Problems**.** *Rensselaer Polytechnic Institute Troy*

Oppen, J. och Lokketangen, A., 2006, Arc routing in a node routing environment. *Computers & Operations Research*, 33 (2006), pp. 1033–1055

Ponce de Leao, M.T. och Matos, M.A., 1999. Multicriteria distribution network planning using simulated annealing. *International Transactions in Operational Research*, 6, pp. 377-91

Ralphs, T.K., Kopman, L., Pulleyblank, W.R. och Trotter, L.E., Jr., 2001. On the Capacitated Vehicle Routing Problem. *Mathematical Programming Series* B 94 (2003), pp. 343

Taillard, E., Badeau, P., Gendreau, M., Guertin, F. och Potvin, J., 1997. A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows. *Transportation Science* 31(1997), pp. 170-186.

Whitley, D. 1994. A Genetic Algorithm Tutorial. *Computer Science Department Colorado State University.*

Wrangtorp, M. 2006. Verbal Modelldokumentation. *Establish*