Ruteplanlægning i praksis

Allan H. Rasmussen

Kgs. Lyngby 2004
IMM-THESIS-2004-51

Ruteplanlægning i praksis

Allan H. Rasmussen

Kgs. Lyngby 2004

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-THESIS: ISSN 1601-233X

Abstrakt

Gennem tiden har planlægning af produktion og distribution af varer været en proces, der har involveret en stor mængde ressourcer og et stort antal mennesker. Planlægningen har ofte været af manuel karakter og ofte uden fundament i videnskaben. Ved hjælp af operationsanalyse er det blevet muligt at beskrive mange problemstillinger indenfor disse områder og udregne optimale løsninger, eller vha. heuristikker at give hurtige bud på løsninger, der er tæt på den optimale løsning.

Dette projekt omhandler distribution i de særlige tilfælde, hvor distribution og produktion er tæt knyttet til hinanden. Mere specifikt omhandler projektet kombinationen af produktion og distribution af gas i Danmark og hvordan en løsning til et sådant problem kan findes. Projektet tager udgangspunkt i en case placeret i Danmark, men problemstillingen stammer oprindeligt fra Sverige. Specielt for produktion af gas gælder at produktionen skal holdes kørende, hvorved der udover en maksimal produktion også er en minimal produktion. Problemet vil blive forsøgt løst med en matematisk model der giver den optimale løsning. Da dette er meget tidskrævende kan det ikke forventes at denne løsningsmetode er tilfredsstillende ved daglig brug, hvorfor hurtigere løsningsmetoder baseret på metaheuristikker, specielt tabu søgning, også anvendes. Hovedvægten i opgaven er lagt på tabu søgning.

Nøgleord: Ruteplanlægning, Produktionsplanlægning, Matematisk model, Metaheuristik, Tabu søgning, Dynamisk indstilling af parametre.

Abstract

Production planning and distribution of goods has for a long time been a process, which involves a great deal of resources and people. Planning has often been done manually and with no or insufficient foundation in science. With Operational Research it is possible to describe and solve many cases in these areas and return an optimal solution, or by heuristic methods to produce near-optimal solutions.

This project deals with distribution in the special case where distribution and production are integrated. More specifically it deals with the combination of production and distribution of gas in Denmark and how a solution to such a problem can be found. The problem originates from Sweden but is presented as a case located in Denmark, due to the availability of data. Production of gas is special because the production must run continuously, which is why a minimum production capacity must be met. Furthermore a maximum production capacity is also given. I will try to solve the problem by making a mathematical model and find an optimal solution. Finding an optimal solution is normally very time consuming. Therefore real-time decision support would instead focus on near-optimal solutions, which is why faster solutions based on metaheuristics, especially tabu search, will be used also. The main focus in the project is on tabu search.

Keywords: multi-depot vehicle routing problem, metaheuristics, tabu search, dynamical adjustment of parameters.

Forord

Jeg vil gerne takke Jørgen Wanscher for at gøre mig opmærksom på Transvisions eksamensprojekt og lede mig i retning af Jesper Larsen som vejleder.

Jeg vil især takke Jesper Larsen for at være min vejleder, altid have tid og gode råd. Jeg går altid fra Jespers kontor med god selvtillid og et optimistisk syn på sagerne, selvom jeg ikke altid har haft begge dele når jeg trådte ind på kontoret.

Ligeledes vil jeg takke Jakob Birkedal Nielsen fra Transvision for samarbejdet med Transvision og for at udbyde eksamensprojektet.

Allan H. Rasmussen

Indholdsfortegnelse

Figurliste

1 Indledning

I dette afsnit vil jeg give en introduktion til problemet, der bliver betragtet i denne opgave og motivationen for dette. Selve projektet er et samarbejde mellem virksomheden Transvision og jeg. Problemstillingen er formuleret af Transvision, baseret på et problem fra den virkelige verden, nærmere bestemt Sverige, men siden hen er problemstillingen tilpasset danske forhold. Et datasæt er desuden udleveret og alle beregninger er baseret på dette datasæt.

Der er både et generelt problem og et specifikt problem, hvor alt data er velkendt og problemet er statisk.

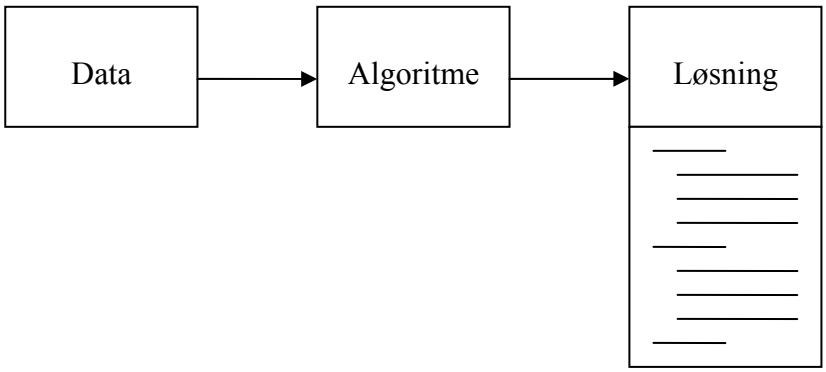
Det generelle problem omhandler løsning af et system, hvor distribution og produktion er tæt sammenknyttet, uden at man på forhånd kender antallet af produktionssteder, transportmulighederne eller ordremængden. Dette kan svare til en situation i den virkelige verden og det er derfor relevant, om min løsningsmetode kan håndtere en sådan situation. Typisk vil produktionsstederne og transportmulighederne være fastlagt, men ordrerne vil være forskellige fra gang til gang. Grundlæggende er problemet derfor dynamisk.

Det specifikke problem omhandler løsningen af et udleverede datasæt. Dette datasæt indeholder 4 produktionssteder også kaldet terminaler, 60 vogne og 250 ordrer.

Den grundlæggende ide i dette projekt er lave en avanceret algoritme, der ud fra nogle fastlagte regler finder en god løsning. Dette vil jeg opnå ved at benytte en metaheuristik. Transvision benytter selv hovedsageligt grådige algoritmer og sådanne algoritmer har typisk et meget fastlagt handlingsmønster. Et eksempel på en grådig algoritme er Dijkstra's algoritme til at finde korteste veje. Her vælges altid den knude med kortest længde (mindste vægt) fra roden. Når man benytter en grådig algoritme, håber man at finde frem til det globale minimum, ved at vælge det lokale minimum i enhver situation. I Dijkstra's tilfælde kan man bevise at resultatet altid er den optimale løsning, men det er ikke altid tilfældet for en grådig algoritme. Især komplicerede problemstillinger, som den behandlet i denne opgave, vil ikke kunne løses til optimalitet vha. en grådig algoritme.

Om en grådig algoritme er succesfuld eller fejler, kan let illustreres ved den klassiske opgave med møntoptælling. Opgaven går ud på at optælle et beløb i mønter, ved at benytte så få mønter som muligt. Den grådige algoritme vil her altid vælge den største mønt der er mulig. Lad os optælle 17 kr. Har vi et system, hvor der findes mønterne 1kr, 5kr og 10kr vil 17kr blive optalt med 4 mønter, nemlig 10kr + 5kr + 2 gange 1kr. Dette er optimalt. Har vi derimod et system, hvor der findes 1kr, 8kr og 10kr vil 17kr blive optalt med 8 mønter, nemlig 10kr + 7 gange 1kr. Dette er ikke optimalt da 2 gange 8kr + 1kr også giver 17kr, men kun med 3 mønter.

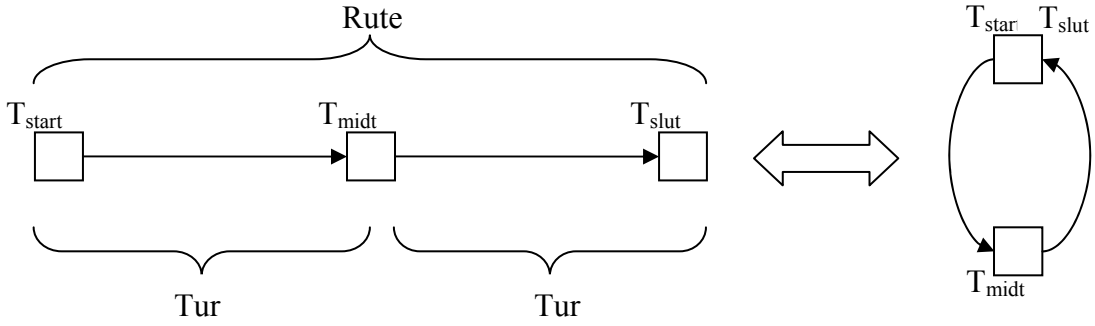
I denne opgave er der visse fundamentale begreber jeg allerede nu vil fastslå. Ud fra relevant data om produktionssteder, transportmuligheder og efterspørgsel ønskes en løsning på produktions- og distributionsproblemet. Denne løsning skal indeholder en liste af vogne, der hver kører en udspecificeret rute, og en angivelse af hvor meget hvert produktionssted skal producere.



Figur 1 – Fra data til algoritme til løsning

En vogn kan kun køre en rute per dag, så vogn og rute beskriver det samme i forhold til løsningen. En rute er en liste af produktionssteder og kunder, der skal besøges i løbet af dagen. En rute starter ved vognens tilhørssted og slutter samme sted ved rutens afslutning. En vogns tilhørssted er i denne opgave altid et produktionssted. Hver rute kan besøge flere produktionssteder end rutens start- og slutsted. En rute består af en eller flere ture.

En tur beskriver rejsen mellem to produktionssteder, således at en rute der kører fra startproduktionsstedet til et produktionssted og tilbage til slutproduktionsstedet indeholder to ture, som illustreret på nedenstående Figur 2.



Figur 2 – Illustration af en rute med 2 ture
Da start- og slutsted altid er identisk bliver ruterne løkkeformet

Mellem to produktionssteder kan vognen stoppe ved en eller flere kunder, for at opfylde deres ordrer. Et besøg, der både kan være et produktionssted eller en kunde, kaldes for et knudepunkt. Figur 4 har f.eks. 6 knudepunkter, hvoraf 3 er produktionssteder og 3 er kunder.

Hvis alle ordrer i løsningen bliver opfyldt i løbet af en dag er opgaven løst, men dette skal helst gøres på en måde, der minimerer omkostningerne.

2 Beskrivelse af problemet

Nedenunder er den originale problembeskrivelse jeg modtog ved projektets start. Bemærk at ordene ’opgave’ og ’ordre’, ’bil’ og ’vogn’ samt ’produktionssted’ og ’terminal’ parvist beskriver det samme igennem hele rapporten.

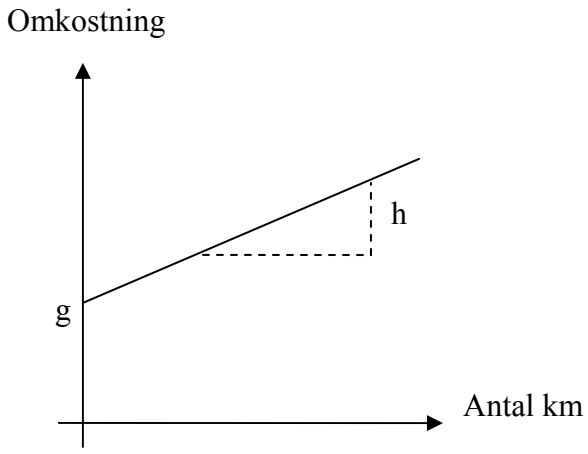
2.1 Problemformulering

Opgaverne:
Opgaverne består af en mængde (kg) samt en lokalitet (x, y). Det antages at losse-tiden er identisk for alle opgaver: En fast stoptid (s minutter) + en variabel tid (t minutter per kg). Opgaverne er af en sådan størrelses-orden, at der typisk er 1-5 opgaver pr. tur. Der kan være flere opgaver til samme kunde (lokalitet).

Bilerne:
Bilerne er til rådighed i et givet tidsrum (varierer fra bil til bil) og har en given lasteevne (varierer fra bil til bil). Bilerne har et tilhørsforhold til et produktions-sted (skal starte dagen her og slutte dagen her), men alle biler kan i princippet betjene alle opgaver. Bilerne har en omkostnings-struktur (g kroner for at starte op + h kroner pr. km).

Produktions-stederne
Et produktions-sted har en max. kapacitet (kg/dag), en min. kapacitet (kg/dag), samt en produktions-omkostning (kr/kg). Det antages at laste-tiden er den samme for alle produktions-steder: Desuden er der en fast stoptid (u minutter) + en variabel tid (z minutter per kg). Produktions-stederne har ubegrænset åbningstid og produktions-stederne kan have vilkårligt mange biler til tankning på én gang

Fælles for losse-tiden, laste-tiden og omkostningsstrukturen er, at der er en initiel omkostning plus en variabel omkostning, som illustreret på Figur 3.



Figur 3 – Initiel omkostning plus variabel omkostning

Beskrivelsen skal udvides på nogle punkter, da en revurdering af problemet er foregået i løbende dialog med Transvision. Efter den første præsentation af min opfattelse af problemet, udspecificeret ved hjælp af den matematiske model, stod det klart at Transvision havde en anden forestilling af problemet end jeg havde. Det førte til følgende punkter:

Opgaverne:

- En opgave må ikke splittes i flere portioner.

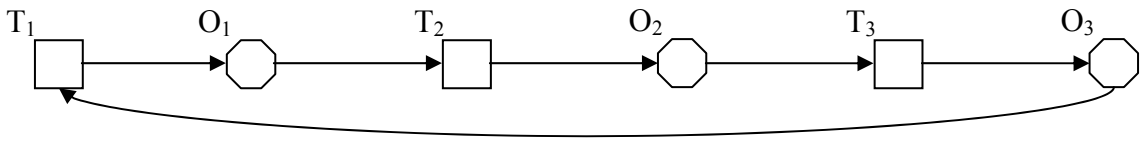
Bilerne:

- En bil må gerne besøge et eller flere produktionssteder i løbet af dagen, dvs. den er ikke begrænset til kun at besøge produktions-stedet hvor den starter og slutter.
- En bil skal være tom inden et nyt produktionssted besøges, da gassen skal kunne spores til produktions-stedet og da chaufførerne foretrækker denne praksis.

Konsekvenserne af disse tilføjelser var mere omfattende, end jeg umiddelbart havde forestillet mig. Problemet går fra et forholdsvis velbeskrevet område i litteraturen til et næsten ubeskrevet blad. Dette blev først klart da den matematiske model skulle tilpasses de nye krav. En sand eksplosion af variable og begrænsninger var resultatet.

2.2 Litteraturstudie

Der er skrevet meget litteratur om tabu søgning, også i relation til ruteplanlægning. Desværre har jeg ikke kunnet finde noget, der beskriver præcist vores udvidede problemstilling. Den litteratur der kommer tættest på, er forskellige varianter af Multi-Depot Vehicle Routing Problem (MDVRP). I MDVRP haves typisk de samme begrænsninger, som vi har, dog må en vogn ikke foretage flere ture men kun en hel rute. Som et specialtilfælde af MDVRP findes full-load, hvor hver vogn kan foretage flere ture, men kun med en ordre per tur. Denne ordre skal desuden have en forespørgsel på et antal fulde vognlæs, hvilket resulterer i en rute af typen som vist på Figur 4. Her betegner T et produktionssted og O en kunde.



Figur 4 – Full-load rute

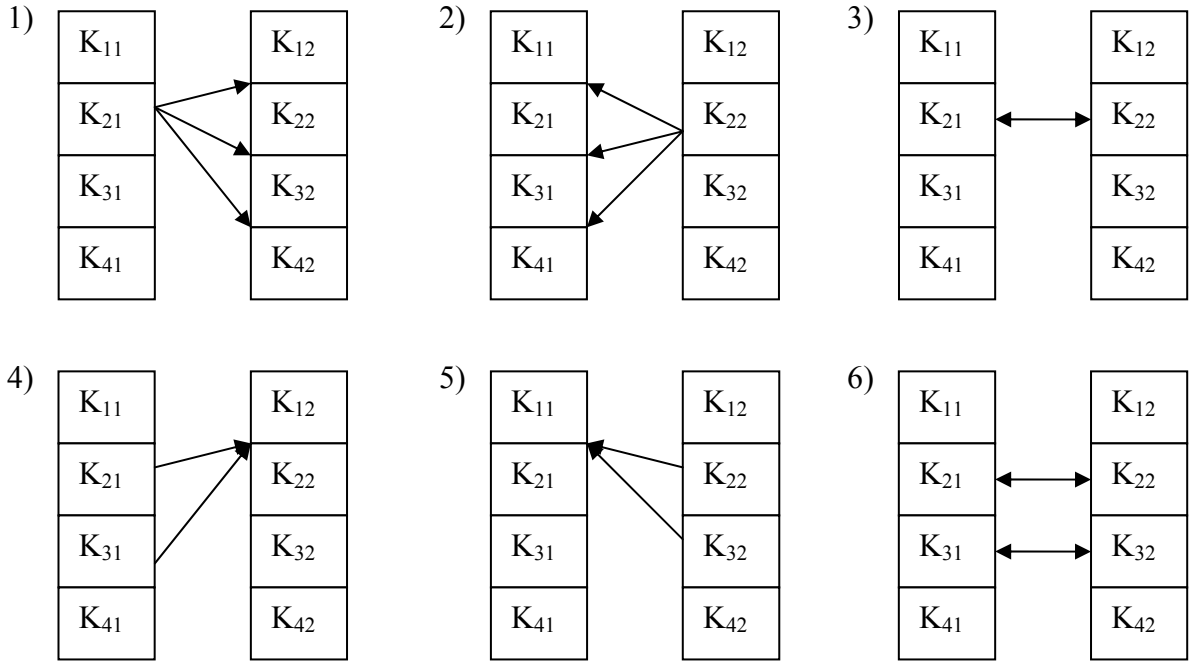
Herunder vil jeg beskrive MDVRP generelt og specielt full-load MDVRP.

Som beskrevet af Renard et al. [Ren96] er Multi-Depot Vehicle Routing Problem (MDVRP) NP-hårdt (dvs. man antager det kan ikke løses i en polynomiel tid) og meget svært at løse til optimalitet selv for små instanser. Kun to eksakte algoritmer er fremsat, begge af Laporte et al. [Lap84][Lap88] og ingen af disse kan løse store problemer (n>50). Derimod findes der mange algoritmer baseret på heuristikker til løsning af MDVRP. Tilføjes produktionsplanlægningen til MDVRP, haves et kompliceret problem, der ikke kan løses til optimalitet med de størrelser, der er angivet i problembeskrivelsen.

Kigger vi nærmere på den matematiske model i [Ren96] er der mange ligheder, men også følgende forskelle, i forhold til modellen i denne opgave. Matricen for kørselstider er i deres tilfælde begrænset til det symmetriske tilfælde, hvor man i min problemstilling kan have forskellige tider og omkostninger fra a til b og fra b til a. De benytter desuden identiske vogne. Begge disse forskelle er uden større betydning og udgør ikke et problem i at formulere en udvidet model. Det springende punkt er udelukkende, om mere end en tur og terminal er tilladt. Heuristikken de benytter er baseret på tabu søgning og desuden nævner de også strategier til intensification og diversification, der henholdsvis fokuserer tabu søgningen i et lovende område eller spreder den til uudforskede områder. De benytter forholdsvis avancerede ombytninger i deres 3 procedurer, 1-route, 2-route og 3-route. Hvert af disse navne refererer til antallet af ruter, der indgår i selve proceduren. I f.eks. 2-route haves en sekvenser af fire knudepunkter K_{1r}, K_{2r}, K_{3r} og K_{4r} fra to ruter r=1 og 2. Herfra forsøges 6 træk, så længe der ikke flyttes en terminal og løsningen ikke bliver ulovlig.

1) K_{21} indsættes mellem K_{12} og K_{42}
2) K_{22} indsættes mellem K_{11} og K_{41}
3) K_{21} og K_{22} ombyttes
4) K_{21} og K_{31} indsættes mellem K_{12} og K_{22}
5) K_{22} og K_{32} indsættes mellem K_{11} og K_{21}
6) K_{21} og K_{31} ombyttes med K_{22} og K_{32}.

Disse 6 træk er illustreret på Figur 5 og er en undergruppe af λ-interchange proceduren af Osman [Osm93].



Figur 5 – 6 forskellige træk benyttet i [Ren96]

[Ren96] nævner case studies for MDVRP for blandt andet levering af færdiglavet måltider, kemiske produkter, læskedrik, maskiner, industriel gas, petroleumsprodukter osv. så den oprindelige problembeskrivelse synes at ligge under MDVRP området. Flere studier har vist at der kan spares betydelige summer ved at benytte teknikker til optimering. Heuristikken udviklet i [Ren96] slår tidligere heuristikker i 20 ud af 23 tilfælde og vinder i alle 23 tilfælde ved parametre indstillet efter problemet.

Full-load MDVRP som beskrevet i [Sum95] er baseret på Clarke og Wright ombytningsprocessen, hvor to ture eller ruter sammenlægges til en hvis det genererer en besparelse. En lovlig initiel løsning findes, hvorefter veje ombyttes efter metoden beskrevet af Clarke og Wright, hvis det genererer en besparelse, indtil ingen ombytninger giver en besparelse. Det svarer til en lokal søgning, hvor et lokalt minimum ikke kan undslippes. Løsningens lovlighed bibeholdes hele tiden. Alle vogne er identiske, efterspørgslen opgøres i fulde vognlæs og vognene er desuden begrænset i antal ture per dag.

Både [Ren96] og [Sum95] refererer specielt til 4 tidligere MDVRP heuristiske algoritmer. Den første af disse er af Tillman og Cain og benytter den klassiske algoritme af Clarke og Wright, hvor ordrer bliver tildelt den nærmeste terminal og vejene mellem disse derefter kan ombyttes, hvis det genererer en besparelse. Algoritmen er en konstruktionsalgoritme.
Den anden algoritme er af Gillett og Johnson og udvider en sweep algoritmen af Gillett og Miller, hvor ordrerne bliver tildelt en terminal og for hver af disse terminaler løses VRP (dette er også kendt som cluster-first route-second). For at forbedre løsningen vælges de ordrer, der ligger i regionen mellem to terminaler og en omfordeling forsøges, med skiftevis hver af terminalerne virkende, som en magnet på ordrerne.
Den tredje heuristik er af Wren og Holliday, hvor ordrerne igen bliver tildelt den nærmeste terminal, men desuden også sorteret efter vinkel i polære koordinater. Dermed kan de udføres i den rækkefølge, der giver mindst distance. Ordrerne kan desuden byttes mellem terminalerne.
Den fjerde og sidste er af Golden et al. og er udvidelser af heuristikkerne fra Tillman og Cain, samt Gillett og Johnson, med henblik på større problemer. Resultater fra Tillman og Cain er sammenlignelige med Gillett og Johnson. Der findes flere heuristikker indenfor området, men disse vil jeg ikke komme nærmere ind på.

Der findes desuden en masse litteratur om Vehicle Routing Problems with Time Windows (VRPTW). VRPTW er ikke umiddelbart relevant for denne opgave, men en artikel af Cordeau et al. [Cor01] har et interessant element. I [Cor01] benytter de tabu søgning, diversification og desuden tillader de også ulovlige løsninger, men tildeler dem en strafværdi. Det sidste er ikke en ny ide, men problemet har altid været hvilken strafværdi man skulle vælge. Det nye er at strafværdien er dynamisk og afhænger af flere faktorer bla. den totale overskridelse af lasteevne, tidsforbrug og tidsvinduer. Dette er især nyttigt i tunge problemer, hvor begrænsningerne er meget svære at opfylde. Algoritmen kan lettere udforske nye områder af løsningsrummet og selvom løsningen bliver ulovlig, vil den dynamiske straf hele tiden lede løsningen hen mod et lovligt område.

Da ingen af de artikler jeg har læst, omhandler den specifikke problemstilling jeg står overfor, kan jeg ikke umiddelbart bruge nogen af artiklerne direkte. Jeg kan derimod bruge delelementer, især med hensyn til tabu søgning og de erfaringer artiklerne beskriver.

3 Analyse af problemstillingen

For at få et overblik over problemet, begyndte jeg tidligt at formulere en klassisk matematisk model, ud fra den oprindelige problembeskrivelse. Målet med en matematisk model, er at finde en optimal løsning, så et sammenligningsgrundlag kan etableres, med henblik på de løsninger en heuristisk algoritme kan frembringe. Desuden giver en matematisk model en mere nøjagtig beskrivelse og definition af problemstillingen.

3.1 Model

Dette er et forsøg på at lave en model, der opfylder de krav, der er stillet i problemformuleringen. En mængde ordrer skal opfyldes vha. nogle vogne, der kan køre frit mellem terminalerne og ordrerne. Vognene er begrænset af deres lasteevne og rådighedstid. Terminalerne er begrænset af deres minimale og maksimale produktionsmængde.

Mængder:

V : Vogne (indeks i)
O : Ordrer, inkl. Terminaler der opfattes som ordrer, såkaldte terminal-ordrer (indeks j,a,b)
T : Terminaler (indeks t)
V_t : Vogne tilhørende terminal t (indeks i_t)
To : Terminal-ordrer (indeks k)

Parametre:

P_t : Produktion for terminal t
c_t : Produktionsomkostninger for terminal t
$Pmin_t$,$Pmax_t$: Min. og max. produktionskapacitet for terminal t
D_{ab} : Distance fra ordre a til ordre b
d : Omkostning per distance
S_i : Startomkostning for vogn i
w_j : Vægt af ordre j
W_i : Lasteevne for vogn i
s : Fast stoptid
t : Tidsforbrug for lastning og losning per vægtenhed
T_{ab} : Kørselstid fra ordre a til ordre b
R_i : Rådighedsperiode for vogn i
M : Meget stor konstant
K : Max. antal terminal-ordrer

Beslutningsvariable:

x_{ij} : 1 hvis vogn i servicerer ordre j, 0 ellers
y_{iab} : 1 hvis vogn i kører direkte fra ordre a til ordre b, 0 ellers
z_i : 1 hvis vogn i benyttes, 0 ellers

Målfunktion:

$$Minimer \quad \sum_t c_t P_t + \sum_i \sum_a \sum_b y_{iab} D_{ab} d + \sum_i S_i z_i \tag{0.1}$$

Begrænsninger:

$$P_t = \sum_{i_t} \sum_j w_j x_{i_t j}, \quad \forall t \tag{0.2}$$

$$Pmin_t \le P_t \le Pmax_t \quad , \forall t \tag{0.3}$$

$$\sum_i x_{ij} = 1 \quad \forall j, j \ne k \tag{0.4}$$

$$\sum_j w_j x_{ij} \le W_i \quad , \forall i \tag{0.5}$$

$$s\sum_j x_{ij} + t\sum_j w_j x_{ij} + \sum_a \sum_b T_{ab} y_{iab} \le R_i \quad , \forall i \tag{0.6}$$

$$\sum_{b:b\ne a} y_{iab} = x_{ia} \quad , \forall i, \forall a \tag{0.7}$$

$$\sum_{a:a\ne b} y_{iab} = x_{ib} \quad , \forall i, \forall b \tag{0.8}$$

$$\sum_{a\in S} \sum_{b\notin S} y_{iab} \ge z_i \quad , S \in N, S \ne \emptyset, \forall i \tag{0.9}$$

$$\sum_a \sum_b y_{iab} \le M z_i \quad \forall i \tag{0.10}$$

$$x_{ik} \le x_{i(k-1)} \quad \forall i, \forall k, 2 \le k \le K \tag{0.11}$$

$$x, y, z \in \{0,1\} \tag{0.12}$$

Udover målfunktionen (0.1) er der 11 sæt af begrænsninger i den matematiske model.

Kigger vi først på formel (0.1) er denne opdelt i en produktionsdel ($\sum_t c_t P_t$) og en transportdel ($\sum_i \sum_a \sum_b y_{iab} D_{ab} d + \sum_i S_i z_i$).

I produktionsdelen benyttes formel (0.2), der direkte beskriver den producerede mængde for hver terminal. I transportdelen af formel (0.1) er der en kørselsomkostning, samt en startomkostning ($\sum_i S_i z_i$) for hver vogn der benyttes.

Modellen minimerer med andre ord kun omkostningerne. I formel (0.3) sikres at min. og max. produktionen overholdes, så de respektive kapaciteter ikke overskrides. Formel (0.4) sikrer at efterspørgslen opfyldes, dvs. alle ordrer serviceres undtagen terminal-ordrer. Formel (0.5) og (0.6) omhandler begge egenskaber ved de benyttede vogne. Formel (0.5) sikrer at ordrerne, der tildeles en vogn, ikke overskrider dennes lasteevne. Formel (0.6) sikrer, at det samlede tidsforbrug ved udførslen af disse ordrer ikke overskrider vognens samlede rådighedsperiode. Formel (0.6) består af 3 led på venstre side, hvor første led er den faste stoptid ved hver ordre, inkl. ved terminalen, der også opfattes som en ordre. Andet led er det vægtafhængige tidsforbrug ved lastning og losning,

mens tredje led er kørselstiden. Formel (0.7), (0.8) og (0.9) er alle TSP[1] begrænsninger. De sørger henholdsvis for at vognen forlader en ordre en gang, ankommer til en ordre en gang og at usammenhængende løsninger frasorteres (cutset), så kun en samlet rute er tilbage. I formel (0.10) sammenkobles y og z, således at y kan antage værdier forskelligt fra nul hvis z=1, dvs. vognen kan kun køre hvis den bliver benyttet. Formel (0.11) sikrer at en terminal-ordre ikke udføres, hvis ikke dens forgænger er udført og at det samlede antal terminal-ordrer per vogn ikke overskrider en given grænse. Formel (0.12) sikrer at x, y og z er binære variable.

Ovenstående matematiske model er baseret på de tidligere listede krav (markeret med •) og ved hvert krav er tilkendegivet om kravet er opfyldet (markeret med +) og i så fald af hvilken formel i modellen. Enkelte krav er ikke relevante som begrænsninger men mere som en beskrivelse af situationen, hvorfor de ikke er kommenteret.

Beskrivelse af problemstillingen

Opgaverne:

- • *Opgaverne består af en mængde (kg) samt en lokalitet (x,y)*
- \+ Generelt opfyldt af modellen, da hver ordre er beskrevet ved en vægt og både distance og kørselstid udregnes vha. koordinater.
- • *Det antages at losse-tiden er identisk for alle opgaver: x minutter + y minutter pr. kg*
- \+ Indgår i formel (0.6) der opfylder det samlede tidsforbrug for hver vogn
- • *Der kan være flere opgaver til samme kunde (lokalitet)*
- \+ Modellen burde uden problemer kunne udføre flere ordrer ved samme lokalitet, både betjent af samme vogn, men også af flere vogne.

Bilerne:

- • *Bilerne er til rådighed i et givet tidsrum (varierer fra bil til bil)*
- \+ Vognenes rådighedsperiode er opfyldt af formel (0.6)
- • *Bilerne har en given lasteevne (varierer fra bil til bil)*
- \+ Vognenes lasteevne er opfyldt af formel (0.5)
- • *Bilerne har et tilhørsforhold til et produktions-sted (skal starte dagen her og slutte dagen her)*
- \+ Vognenes tilhørsforhold er opfyldt af TSP begrænsningerne (0.7 til 0.9) da hver TSP tour er en lukket kreds, der starter og stopper i samme punkt, dvs. terminalen.
- • *Alle biler kan i princippet betjene alle opgaver*
- • *Bilerne har en omkostnings-struktur (g kroner for at starte op + h kroner pr. km)*
- \+ Opfyldt af anden del af målfunktionen (0.1)

Produktions-stederne

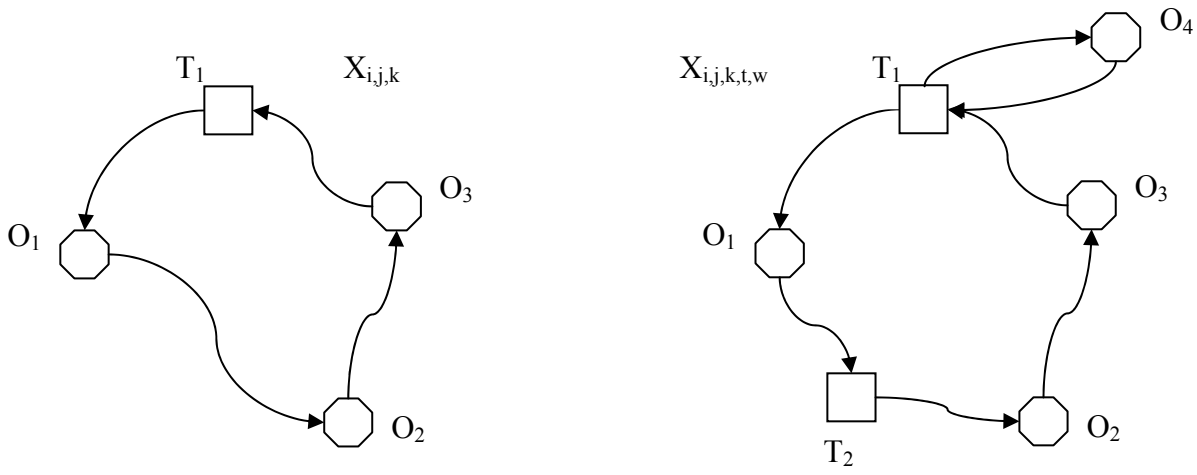
- • *Et produktions-sted har en max. kapacitet (kg/dag) en min. kapacitet (kg/dag) samt en produktions-omkostning (kr/kg)*
- \+ Min. og max. produktionskapacitet er opfyldt af formel (0.3) mens produktionsomkostningerne indgår som første del af målfunktionen (0.1).
- • *Det antages at laste-tiden er den samme for alle produktions-steder: u minutter + z minutter pr. kg*
- \+ Indgår i formel (0.6) der opfylder det samlede tidsforbrug for hver vogn

[1] Travelling Salesman Problem (TSP)

- *Produktions-stederne har ubegrænset åbningstid*
- *Produktions-stederne kan have vilkårligt mange biler til tankning på én gang*

Havde krav som *Alle biler kan i princippet betjene alle opgaver* væres anderledes, så kun visse biler kunne betjene visse opgaver, ville det kun give anledning til simple begrænsninger, der ikke ville give anledning til at løsningen blev betydeligt sværere.

Som tidligere beskrevet blev visse af forudsætningerne for problemet præciseret. En af ændringerne er, at en vogn gerne må besøge flere terminaler, både dens startterminal og andre, i løbet af en rute, blot der sluttes ved samme terminal, som der blev startet fra. Specielt dette krav øger kompleksiteten af problemet, da man udover de allerede bestemte variable også skal holde styr på hvilken tur vognen udfører på nuværende tidspunkt og hvorfra dens last stammer. På Figur 6 illustrere jeg hvordan stigningen af variable ser ud og de mere komplekse ruter, der kan fremkomme. Bemærk især stigningen af indekserne til X. Hvor der før kun var en rute at holde styr på og om vogn i kørte mellem ordre j og k eller ej, er der nu langt mere at holde styr på. Udover ruten er der 3 ture, der hver kan tage udgangspunkt i forskellige terminaler. For hver tur skal vognens maksimale lasteevne overholdes, modsat for hver rute før. I alt skal man nu holde styr på om vogn i kører mellem ordre j og k stammende fra terminal t på tur w.



Figur 6 – Antal variable ved en simpel og en kompleks rute

Selvom en matematisk model kunne formuleres med de nye krav, ville det ikke være muligt at finde en løsning til denne model, ifølge mine bedste vurderinger. Desuden ville modellen blive for kompliceret til at give overblik, især for udenforstående der læser rapporten. Modellen kunne så kun opfylde et af dens tre krav – en præcis beskrivelse. Derfor går jeg ikke videre med den viste matematiske model.

Til at finde en optimal løsning, benyttede jeg desuden en anden type matematisk model, baseret på søjle-generering. Her løses et hovedproblem, bestående af produktionen og dækning af efterspørgsel, og et underproblem, bestående af ruteplanlægning. Desværre blev det klart efter næsten en måneds arbejde, at sammenkædningen af netop disse to problemer var utroligt svært rent praktisk, hvorfor også denne model måtte opgives.

Skal jeg give et bud på en matematisk model, der kan beskrive hele problemet, skal modellen opdeles i et set partitioning problem (SPP) og nogle yderligere begrænsninger, se f.eks. [Chu98].

SPP er problemet, hvor hver række skal dækkes præcist en gang, samtidigt med at den samlede omkostning minimeres. Beslutningsvariablen er x_j for hver søjle j. Er $x_j = 1$ er søjle j (med omkostning c_j) med i løsningen og 0 ellers. Det giver følgende model:

$$Minimer \quad \sum_{j=1}^{n} c_j x_j \tag{0.13}$$

$$Begrænset\ af \quad \sum_{j=1}^{n} a_{ij} x_j = 1, \quad i = 1,..,m \tag{0.14}$$

$$x_j \in \{0,1\}, \quad j = 1,..,n \tag{0.15}$$

SPP er benyttet i mange sammenhænge, men er især udbredt indenfor airline crew scheduling, hvor hver række repræsenterer en flyvning uden mellemlanding. Søjlerne repræsenterer mulige rotationer for besætningen. En rotation svarer til det jeg kalder en rute, idet man starter og slutter samme sted, efter undervejs at have opfyldt nogle mål. Til hver rotation er knyttet en omkostning c_j og matricen a_{ij} har $a_{ij} = 1$, hvis flyvning *i* er dækket af rotation *j* og $a_{ij} = 0$ ellers. Målet med airline crew scheduling er at finde den samling af rotationer, således at hver flyvning er dækket af præcist en besætning, med mindst mulig omkostning.

Andre faktorer kan forekomme, så som fagforeningsregler og lignende, der tilføjer yderligere begrænsninger. Disse begrænsninger kaldes 'base constraints' og problemet bliver da til et set covering problem.

For denne opgave bliver SPP masterproblemet, bestående af minimering af omkostningerne ved at opfylde hver kundes efterspørgsel. De resterende begrænsninger bliver til et subproblem.

Beslutningsvariablen er x_{ij}. $x_{ij} = 1$ hvis vogn i kører rute j. Omkostningerne for vogn i på rute j er c_{ij}, der indeholder enhedsomkostninger for produktionen, kørselsomkostninger og startomkostninger. Ordrerne har indeks k, $a_{ijk} = 1$ hvis vogn i på rute j servicere ordre k og 0 ellers. P_{ijt} er produktionen for terminal t for vogn i på rute j. Det giver følgende masterproblem:

$$Minimer \quad \sum_i \sum_j c_{ij} x_{ij} \tag{0.16}$$

$$Begrænset\ af \quad \sum_i \sum_j a_{ijk} x_{ij} = 1 \quad , \forall k \tag{0.17}$$

$$\sum_i \sum_j P_{ijt} x_{ij} \geq P_{min_t} \quad , \forall t \tag{0.18}$$

$$\sum_i \sum_j P_{ijt} x_{ij} \leq P_{max_t} \quad , \forall t \tag{0.19}$$

$$x_{ij} \in \{0,1\} \tag{0.20}$$

Hvor (0.16) svarer til (0.13), (0.17) kræver at kundernes efterspørgsel opfyldes og (0.18) og (0.19) sørger for henholdsvis min. og max. produktionsmængde for hver terminal.

Subproblemet består af de yderligere begrænsninger:

- Egenskaber ved vognene overholdes, bla. lasteevne og rådighedstid.
- Ruternes egenskaber jævnfør TRP begrænsninger.

4 Gennemgang af tabu søgning

Tabu søgning er en heuristik, der blev lanceret i midten af 80'erne, i forbindelse med forskning i kunstig intelligens [Lau94]. Siden er det blevet klart, at tabu søgning klarer sig godt indenfor flere områder af optimering, blandt andet ruteplanlægning [Ren96][Glo95]. Tabu søgning består grundlæggende af følgende elementer:

- En initiel løsning
- Et nabolag
- En tabuliste
- Et aspirationskriterium og
- Et stopkriterium.

Indledningsvis etableres en lovlig løsning kaldet den initielle løsning. Denne løsning behøver ikke være god rent målfunktionsmæssigt. Eneste krav er at den er lovlig, dvs. alle begrænsninger stillet i opgaven skal være opfyldt. Man kan også vælge at den initielle løsning må være ulovlig. Det benyttes, hvis det er svært at finde en løsning, som følge af mange eller stramme begrænsninger eller komplicerede nabolag [Cor01]. Der er åbenlyse ulemper ved at tage udgangspunkt i en ulovlig løsning, bla. kan man ikke altid garantere at nå en lovlig løsning, så hvis en lovlig løsning let kan findes, må det være at foretrække.

Det er ud fra den initielle løsning, at tabu søgning trinvist forsøger at forbedre løsningen og derved målfunktionsværdien. Til den trinvise forbedring benyttes et nabolag, der er en samling af løsninger, kaldet naboer, opnået via modifikationer af en oprindelig løsning. Dette er beskrevet i pseudo-koden for tabu søgning på Figur 7. Modifikationerne af en løsning kan f.eks. være en ombytning af ordrer mellem to ruter, hvorved en ny nabo (til løsningen) fremkommer. Den bedste af disse nye naboer kan herefter vælges og et nyt nabolag kan genereres, med udgangspunkt i den valgte nabo. Fortsættes denne proces, ender man på et tidspunkt i et lokalt minimum. I det næste nabolag vil alle løsninger være dårligere end den nuværende. Vælges en af disse, vil den bedste løsning fra det nye nabolag netop være det lokale minimum igen. Algoritmen hænger fast i det lokale minimum.

For at forhindre denne situation forbyder man algoritmen at gå tilbage til allerede besøgte løsninger, eller rettere sagt tidligere benyttede ombytninger, ved at gemme disse i en tabuliste. Desuden forbydes den inverse ombytning altid, da det aldrig kan betale sig at lave to iterationer og ende i præcist samme situation, som man startede i. Tabulisten forhindrer nu effektivt enhver form for cyklisk opførsel, i hvert fald indenfor en vis grænse, men bliver tabulisten for lang, hæmmer det også algoritmen. Algoritmen kan blive hæmmet på to forskellige måder, reduceret bevægelighed og tidsmæssigt forbrug. Ved reduceret bevægelighed er antallet af træk tilføjet til tabulisten så stort, at algoritmens søgning i løsningsrummet bliver påvirket i betydelig negativ retning. Ved tidsmæssigt forbrug er der tilføjet så mange træk til tabulisten, at en søgning i listen tager forholdsvis lang tid. Typisk vil en kombination af disse to negative konsekvenser påvirke algoritmen, hvis tabulisten bliver for lang. Som følge heraf begrænses tabulisten til en passende længde. Typisk benyttes en længde på 7 som udgangspunkt [Lau94], men om det er et fornuftigt valg afhænger af situationen.

Elementer, der tilføjes til tabulisten, kan være af forskellig type. Så længe elementet unikt beskriver hændelsen, der ikke må foregå igen, mens elementet er i tabulisten, giver det mening at tilføje elementet til en tabuliste. Foretages en 1 til 1 ombytning af to ordrer, kan man indsætte følgende elementer i tabulisten:

- De to ordrer
- De to ture ordrerne stammer fra
- De to ruter ordrerne stammer fra
- De to vogne ordrerne stammer fra

Bemærk at kun det første af disse elementer vil forhindre ombytning af de to ordrer igen, indenfor tabulistens længde. De involverede ordrer kan efter nogle iterationer være byttet med andre ordrer til andre ture, ruter eller vogne, hvorefter de to ordrer kunne ombyttes igen indenfor tabulistens længde. Til gengæld kan et element med f.eks. to vogne på tabulisten udelukke en større del af løsningsrummet end blot to ordrer.

Desuden indeholder tabu søgning ofte et aspirationskriterium. Aspirationskriteriet skal modvirke tabulistens hæmmende virkning, især med henblik på spring til uudforskede dele af løsningsrummet. Et eksempel på et klassisk aspirationskriterium, er hvor en løsning bliver valgt hvis det er den bedste løsning hidtil, selvom løsningen er på tabulisten [Lau94], dvs. ombytningen, der fører til løsningen, er på tabulisten. Derimod vælges altid bedste tilladte løsning, der ikke er på tabulisten, hvis man undlader et sådant aspirationskriterium.

Endeligt haves et stopkriterium, der afgør hvornår algoritmen skal terminere. Eksempler på stopkriterier er antal iterationer, antal iterationer uden forbedring, tidsforbrug eller målfunktionsværdi. Ved antal iterationer uden forbedring eller målfunktionsværdi kan det være meget svært at forudsige kørselstiden. Det er aldrig sikkert, at den værdi man har sat bliver opnået. Ved antal iterationer kan det også være svært at forudsige kørselstiden, men det er evt. muligt at anslå kørselstiden, især hvis datasættet svarer til et man har kørt før. Ved tidsforbrug er kørselstiden i sagens natur veldefineret, men det er til gengæld ikke sikkert, algoritmen er kommet tæt på en god løsning. En kombination af stopkriterier kan med fordel benyttes, f.eks. kan man kombinere antal iterationer uden forbedring med et maksimalt tidsforbrug. Her sikrer man at algoritmen stopper, hvis den ikke finder nogle forbedringer og man kender desuden den maksimale kørselstid.

```
1   Find en initiel løsning s
2   Gentag indtil stopkriteriet mødes
3        Generer omegn N(s)
4        For alle løsninger s' i omegnen N(s) beregnes målfunktionsværdien M(s')
5        Hvis M(bedste løsning fra N(s)) < M(hidtil bedste løsning)
6             Så s = bedste løsning fra N(s)
7             Ellers s = bedste tilladte løsning fra N(s) der ikke er på tabulisten
8        Opdater tabulisten
```

Figur 7 – Pseudo-kode for tabu søgning

5 Algoritme

Som beskrevet i et foregående kapitel var det ikke muligt at formulere en matematisk model, der kunne løses med den mængde variable og begrænsninger problemet giver anledning til. Desuden vil der typisk ikke være tid til at finde en optimal løsning i den virkelige verden, selv hvis en sådan kunne findes. Typisk er det irrelevant at tale om en optimal løsning i den virkelige verden, da problemstillingen tit er en forenklet version af virkeligheden. Dette fører til ønsket om en algoritme, der hurtigt giver løsninger tæt på den optimale løsning. Jeg valgte at benytte en algoritme baseret på tabu søgning. Valget af tabu søgning er inspireret af de gode resultater opnået med denne type lokalsøgning indenfor MDVRP og lignende problemer.

Algoritmen består af flere dele som beskrevet tidligere. I dette afsnit vil jeg detaljeret beskrive hver enkelt del af algoritmen.

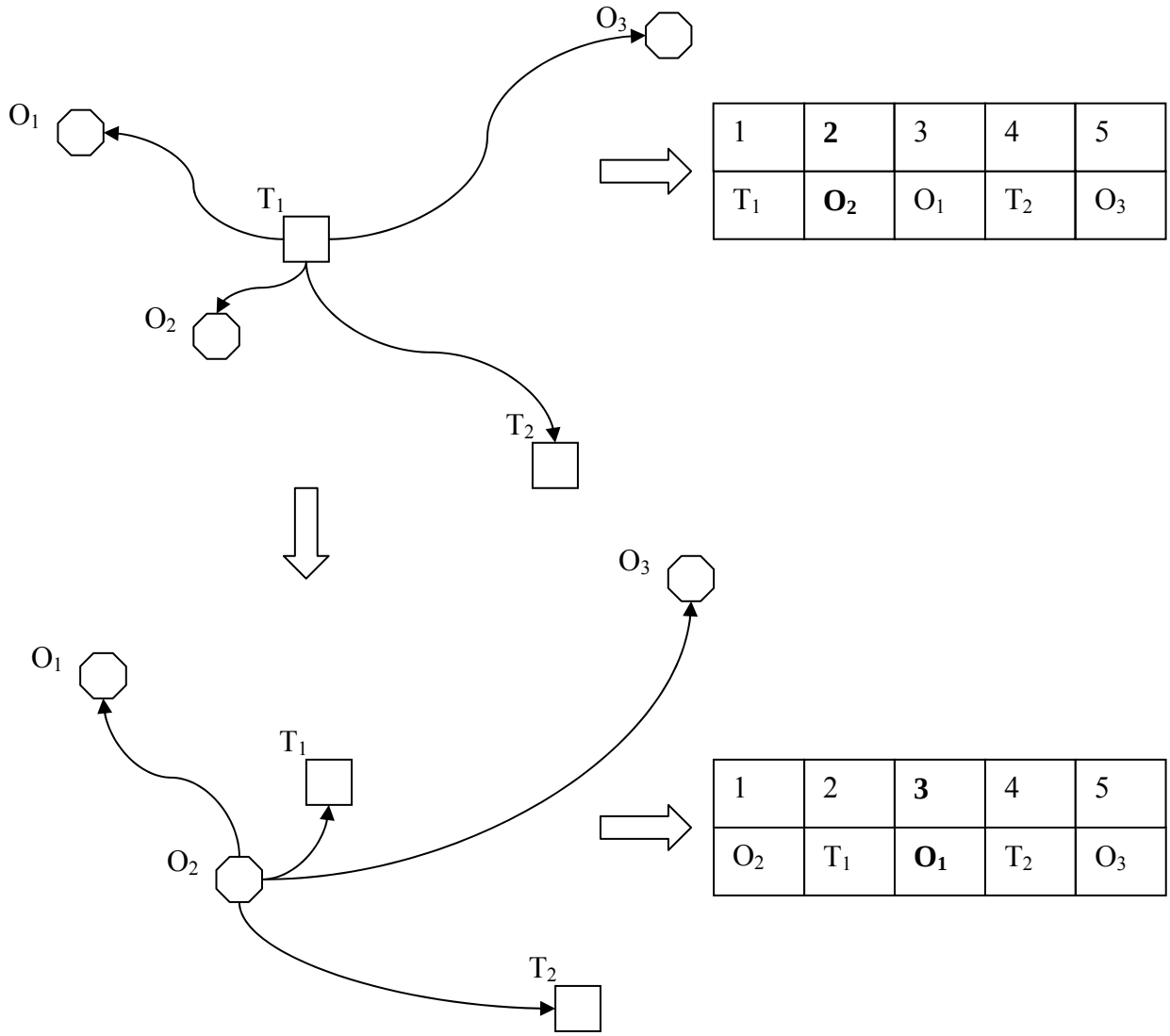
5.1 Initiel løsning

Den initielle løsning er i dette tilfælde en løsning, der opfylder alle begrænsninger stillet i opgaven. En initiel løsning kan som tidligere beskrevet godt være ulovlig, men jeg har valgt kun at acceptere en lovlig løsning, da det er muligt. Desuden sikrer det, at jeg hele tiden har lovlige løsninger i det videre forløb.

5.1.1 Generelt

Den initielle løsning kan tage udgangspunkt i terminalerne, vognene eller ordrerne. Generelt for den initielle løsninger gælder, at jeg prøver at lave den så simpelt som muligt. Hermed mener jeg, at hver vogn ikke får lov til at besøge andre terminaler end den terminal den tilhører. Dette krav glæder kun for den initielle løsning og ikke for den endelige løsning.

Selve opbygningen af en rute i løsningen foregår via en sorteret liste knyttet til hvert knudepunkt. Denne liste indeholder lige så mange elementer som der er knudepunkter. Hvert af disse elementer indeholder et knudepunkt, den korteste kørselsafstand og kørselstid fra knudepunktet man står ved til elementets knudepunkt. Kørselsafstanden og kørselstiden er i denne opgave forudberegnet mellem hvert knudepunkt, hvorfor det ikke giver mening at foretage handlinger mellem to knudepunkter. Starter en vogn ved en terminal, er denne terminal det første knudepunkt, som illustreret via listen øverst på Figur 8. Listen tilknyttet terminalen vil indeholde alle andre terminaler og ordrer (reduceret til 5 på figuren). Antager vi at listen er sorteret efter afstand, vil første element i listen være det knudepunkt, hvor kørselsafstand fra udgangspunktet til knudepunktet er mindst. Første element, der er en uservicerat ordrer, vælges og herfra kan nye ordrer vælges, indtil vognen må returnerer til terminalen. På Figur 8 vælges først ordre O_2 og herefter er det oplagt at vælge O_1, da O_1 er den nærmeste ordre fra O_2. Dette glæder kun, såfremt vognen har nok kapacitet til at servicere både O_2 og O_1. Hvis vognen ikke har nok kapacitet, vil algoritmen forsøge med O_3. Kan denne ordre heller ikke serviceres på samme tur som O_2, må vognen returnere til terminal 1 og en ny tur kan påbegyndes, såfremt den resterende rådighedstid er tilstrækkelig. I princippet kunne vognen også køre til terminal 2, men det komplicerer opbygningen af den initielle løsning unødvendigt. Den opbyggede tur er i øvrigt den korteste tur, der kan konstrueres, indeholdende de valgte knudepunkter i den givne rækkefølge.



Figur 8 – Knudepunkter sorteret efter kørselsafstand eller kørselstid

Til hver terminal er knyttet en mindste og største produktionsmængde. Desuden er vognene også knyttet til terminalerne, så de skal starte og slutte dagen ved samme terminal. For at aktivere en vogn fra en bestemt terminal, skal denne terminal have produktionskapacitet til at opfylde mindste en ordre for vognen. Et af målene for den initielle løsning er, at aktivere så mange vogne som muligt, da nye ruter oftere bliver nedlagt end oprettet i den videre kørsel. Målet kunne være at aktivere alle vognene, men det vil ikke altid være muligt og er umiddelbart ikke muligt i det oprindelige problem fra Transvision. Terminalerne sorteres efter maksimal produktionsmængde i forhold til antallet af vogne, da dette har givet det bedste resultat, især hvis terminalerne sorteres, så den terminal med størst maksimal produktionsmængde i forhold til antal vogne kommer først. Dette er baseret på initielle testkørsler af forskellige muligheder.
Vognenes kapacitet er ikke særlig vigtig, da de fleste vogne har tilstrækkelig kapacitet til at opfylde en gennemsnitlig ordre og en ordre er nok. Der kan konstrueres eksempler, hvor vognenes gennemsnitlige kapacitet er mindre end ordrernes gennemsnitlige mængde, men det udelukker ikke at den initielle løsning finder en lovlig løsning, hvis en sådan eksisterer. Desuden kan hver vogn

køre flere ture, så længe vognens tidsbegrænsning overholdes, hvorved vognen kan transportere en større mængde.

I denne opgave er der to forhindringer for at finde en lovlig initiel løsning. Rådighedstiden i det oprindelige problem er forholdsvis lav og det kræver meget tid at betjene de store ordrer. Den næste forhindring er, at de største ordrer skiller sig ud fra mængden af ordrer, ved at overgå mange af vognenes kapacitet. Det resulterer i at kun få vogne, de største, kan køre de store ordrer. Er de største vogne allerede fyldt med andre ordrer, bliver det hurtigt et problem. Det sidste prøvede jeg at modvirke ved at forfordele de største ordrer, men det gav ikke en bedre løsning.

Da det er vigtigere at få en lovlig initiel løsning end en initiel løsning med god målfunktionsværdi, har jeg fokuseret på tidsforbruget i stedet for kørselsafstanden. Derfor har jeg valgt at benytte en grådig algoritme, til fordeling af ordrer på hver vogn, der forsøger at minimere tidsforbruget. Kigger man på en vogn, der er placeret ved en terminal eller ved en ordre, vil alle andre ordrer og terminaler optræde i en liste sorteret efter kørselstiden. Første element i listen er den ordre eller terminal, man hurtigst kan nå.

5.1.2 5 forskellige løsningsmåder

Jeg har forsøgt mig med 5 forskellige måder at generere den initielle løsning, hvoraf jeg hovedsageligt vil beskrive den, der også bliver benyttet i den færdige algoritme. Hver løsningsmåde svarer til en større ændring fra den forrige måde. Herefter har jeg afprøvet hvilken måde, der giver den bedste løsning og denne har jeg så arbejdet videre med.

De 5 forskellige måder til at finde en lovlig initiel løsning er opnået gennem en iterativ proces baseret på trial-and-error. I de første måder til at generere en initiel løsning forsøgte jeg med en enkel type af fordeling baseret på vognene. Først prøvede jeg at opfylde alle de ordrer, det var muligt på første tur med hver vogn, hvorefter samme procedure gentages med de næste ture, indtil der ikke er flere ordrer. Derefter prøvede jeg at udnytte så meget tid som muligt per vogn, dvs. opfylde så mange ordrer og dermed ture som mulige, indenfor vognens tidsbegrænsning, inden næste vogn fik tildelt ordrer. Begge metoder førte til et forholdsvist stort tidsforbrug og mange ubenyttede vogne, omkring 6-12 stk. ud af de 60. Herefter forsøgte jeg med kun en ordre per vogn, for alle vognene, hvorefter resten af ordrerne blev fordelt. Dette kunne man forvente ville give så få ubenyttede vogne som muligt og det var også sandt. Desværre skal der bruges meget tid på denne løsning, 7 gange mere end oprindeligt til rådighed. Selvom det er vigtigt at aktivere så mange biler som muligt må det ikke ske for enhver pris, især ikke når rådighedstiden for visse vogne overskrider 24 timer per dag!

Efter de 3 eksperimenter med konstruktion af den initielle løsning ud fra vognene, med så ringe resultater at det ikke kan bruges, virkede det som om en helt ny metode skulle afprøves. Det førte til to versioner, hvor fordelingen tager udgangspunkt i terminalerne, som beskrevet i indledningen af dette afsnit. Fordelene ved at tage udgangspunkt i terminalerne og deres produktionsmængde i forhold til antal vogne er, at man lettere kan styre produktionsmængden og fordelingen geografisk. Desuden kan fordelingen styres økonomisk ved at vælge den billigste terminal først. Den første af versionerne baseret på fordeling efter terminalerne er mere eller mindre statisk, da der ikke er noget kriterium for fordeling af terminalerne, udover den rækkefølge man indtaster dem i. Den anden måde er fleksibel mht. rækkefølgen terminalerne vælges i, da terminalerne nu kan sorteres efter et

eller flere kriterier, f.eks. maksimal produktion i forhold til antal vogne og produktionsprisen. Det er denne sidste version, der er den endelige og trin for trin virker den som følgende:

```
1   Terminalerne sorteres  efter maksimal produktionskapacitet per tilknyttet vogn.
2   For hver terminal
3        For hver vogn tilknyttet den valgte terminal
4             For hver ordre (sorteret efter korteste kørselstid)
5                  Hvis ordren er uopfyldt tilføjes ordren til vognen
6                       Hvis turen (vægtbegrænsning) og ruten (tidsbegrænsning) er lovlig efter
                        tilføjelsen af ordren
7                            Så marker ordren som opfyldt
8             Hvis mindst en ordre er tilføjet vognen oprettes turen
9        Hvis mindst en ordre er tilføjet vognen oprettes ruten
10  Returner initiel løsning
```

Figur 9 – Pseudo-kode for initiel løsning

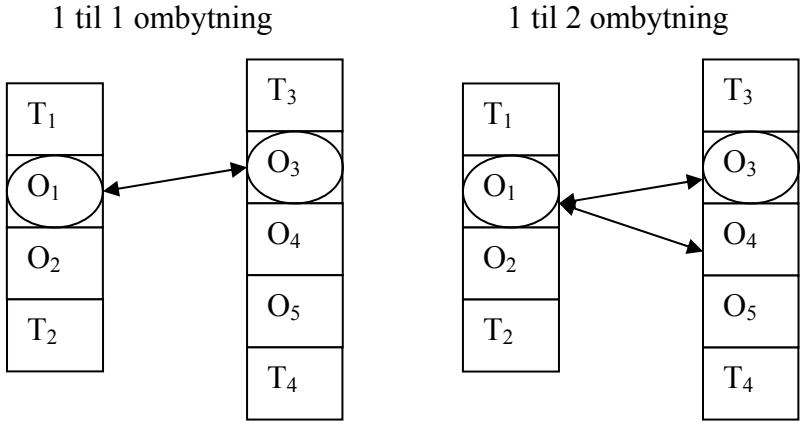
Grunden til jeg har valgt maksimal produktionskapacitet per tilknyttet vogn, som det endelige sorteringskriterium, er baseret på eksperimentelle resultater. Disse resultater kan ikke sammenlignes mht. målfunktionsværdi, men er en vurdering af løsningens kvalitet mht. tidsforbrug og fleksibilitet.

5.2 Tabuliste

Inde i hvert nabolag er der et antal mulige ombytninger. Hver af disse ombytninger skaber en ny nabo i nabolaget. Disse naboer bliver tilføjet en liste, der sorteres efter målfunktionsværdi. Den nabo med bedst målfunktionsværdi vælges, men inden ombytningen foretages bliver det kontrolleret om den er tabu, dvs. er i tabulisten. Er ombytningen tabu må den næste ombytning vælges i stedet. Således fortsættes indtil en ombytning findes, der ikke allerede er i tabulisten. Når denne ombytning er fundet foretages ombytningen og ombytningen tilføjes til tabulisten, så samme ombytning ikke kan ske igen, indenfor tabulistens længde. Skal et nyt element tilføjes en tabuliste af fuld længde bortkastes blot det ældste element.

Elementerne på tabulisten vil typisk være af formen <variabel 1, variable 2>, hvor variablerne kan være ordrer, vogne eller andre datatyper. At et element er tabu betyder ikke, at den specifikke variabel ikke kan ombyttes med andre variable, kun at ombytningen eller overgangen mellem de to specificerede variable er forbudt.

- Ved 1 til 1 eller 1 til 2 ombytning er de to elementer, der tilføjes til tabulisten, illustreret vha. ringe på Figur 10. I begge tilfældet er det henholdsvis O_1 og O_3 der tilføjes. Man kunne godt tilføje 3 elementer i 1 til 2 ombytning, men har jeg ikke valgt at gøre.



Figur 10 – Hvad tilføjes til tabulisten

5.3 Konstruktion af nabolag

Efter en initiel løsning er genereret, konstrueres et nabolag ud fra den nuværende løsning, der i det første tilfælde er den initielle løsning. I dette nabolag genereres alle mulige naboer. Den bedste af disse naboer, som ikke giver anledning til konflikt med tabulisten, vælges som den nye løsning. Herfra konstrueres igen et nyt nabolag og det bedste nabo vælges igen som løsning. Dette forsætter indtil et stopkriterium mødes. Nedenfor vil jeg gennemgå de typer af nabolag jeg har benyttet.

En sammenligning af resultater opnået med de forskellige typer af nabolag er i slutningen af afsnit 5.3.3.

5.3.1 1 til 1 ombytning

Den mest basale konstruktion af et nabolag er 1 til 1 ombytning af knudepunkter [Lau94]. Her tages et knudepunkt fra en rute og ombyttes med et knudepunkt fra en anden rute, baseret på sammenligninger af målfunktionsværdierne for alle mulige ombytninger. Til tabulisten tilføjes de to ombyttede knudepunkter, så disse ikke kan ombyttes med hinanden igen, indenfor tabulistens længde. Det giver ikke mening at ombytte terminaler, hvorfor jeg kun ombytter ordrer. Terminaler bliver behandlet separat i et senere afsnit.
Fordelen ved 1 til 1 ombytning er enkelthed og et stort antal mulige ombytninger, hvilket betyder at mange muligheder bliver undersøgt i løsningsrummet. Typisk for den initielle løsning gælder, at mange af de aktive ture allerede er forholdsvis fyldte, så rent vægtmæssigt kan kun få ordrer indsættes uden en anden fjernes. 1 til 1 ombytning gør netop dette.

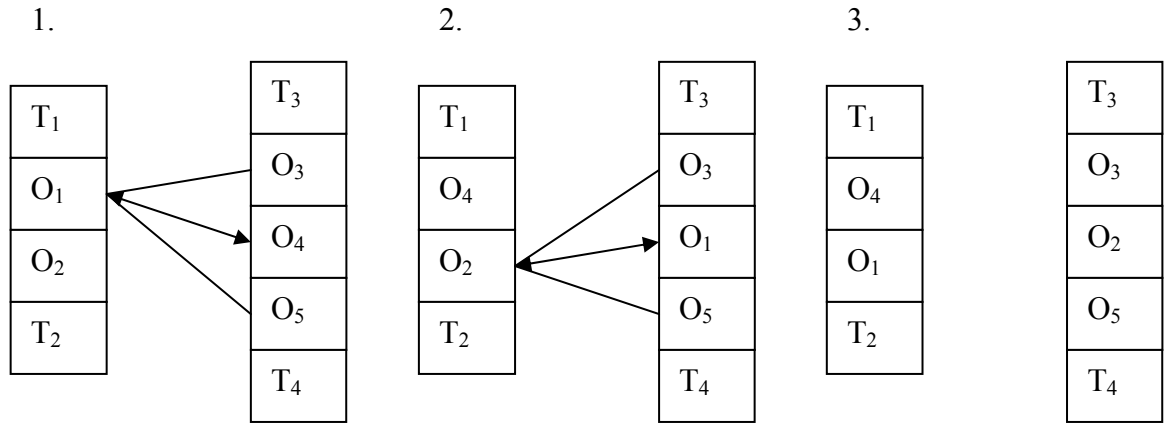
Ulemperne ved 1 til 1 ombytning er at turens og rutens størrelse, dvs. antallet af knudepunkter, ikke ændres. Der er derfor mange mulige løsninger der ikke kan nås, fordi antallet af knudepunkter er låst til antallet fra den initielle løsning. 1 til 1 ombytning udelukker alt for mange løsninger til, at den alene kan finde frem til en god løsning i det generelle tilfælde.

Rent praktisk foregår ombytningen efter nedenstående pseudo-kode:

```
1    Vælg tilfældig rute
2         For hver tur
3              For hver knudepunkt
4                   Hvis knudepunkt er ordre
5                        For hver rute (minus den allerede valgte)
6                             For hver tur
7                                  For hver knudepunkt
8                                       Hvis knudepunkt er ordre
9                                            Foretag ombytning
10                                           Kontroller om løsningen er lovlig
11                                                Hvis ja så gem løsningen
12                                                Hvis nej så ombyt til oprindelig løsning
13                       Vælg ombytningen svarende til bedste gemte løsning
14                            Kontroller om ombytningen er tabu
15                                 Hvis ja så slet ombytningen fra de gemte løsninger og spring til
                                   13
16                                 Hvis nej så foretag ombytningen
17   Returner løsning
```

Figur 11 – Pseudo-kode for 1 til 1 ombytning per ordre

Betragter vi et eksempel med 2 ruter, kun 1 tur hver og henholdsvis 2 og 3 ordrer kan det se ud som følgende (hvor T betegner en terminal og O en ordre):

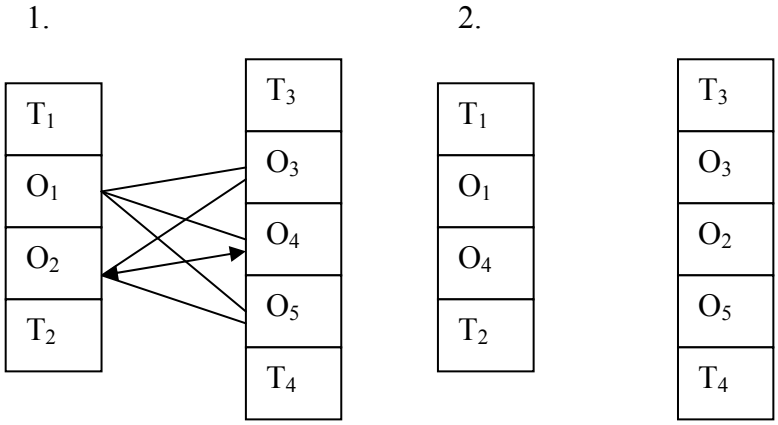


Figur 12 – 1 til 1 ombytning per ordre

I eksemplet på Figur 12 har vi rute 1 med 2 ordrer og rute 2 med 3 ordrer. I eksemplet bliver rute 1 valgt i linie 1 på Figur 11. I situation 1 er koden gennemløbet fra linie 3 til 4 to gange for at finde frem til ordre O_1, linie 7 til 8 to gange for at finde O_3 og vi er nu klar til at gennemløbe linie 9 til 12 første gang. Det er illustreret med stregen mellem O_1 og O_3. Efter dette gentages linie 7 til 12 indtil der ikke er flere ordrer i den nuværende tur på rute 2. Det er illustreret ved de 3 streget mellem de 2 ruter. Måske er ikke alle ombytningerne lovlige, men blandt de lovlige vælges den bedste, såfremt den ikke er på tabulisten. Dette svarer til linie 13 til 16 og er illustreret ved pilespidserne mellem O_1 og O_4.

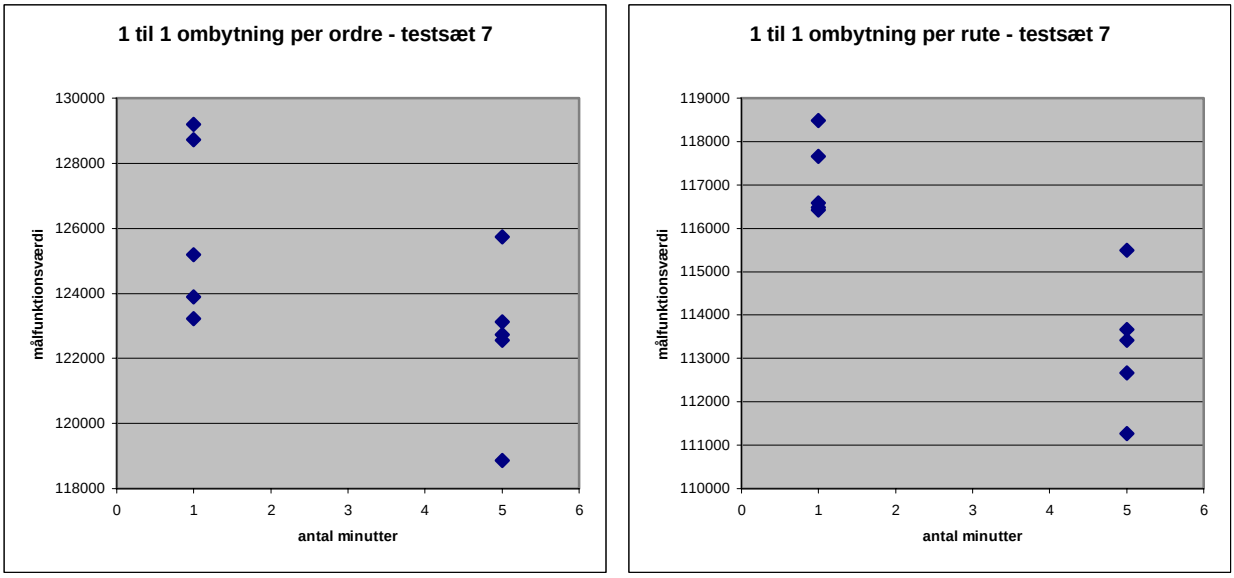
Ombytningen der foretages er O_1 til O_4, da tabulisten ikke indeholdt ombytningen. Ombytningen tilføjes til tabulisten og der springes til linie 3. Efter situation 1 gennemløbes koden igen indtil linie 16. Det er nu O_2 der forsøges ombyttet med de mulige ordrer i rute 2. Det bemærkes at O_1 er en af de mulige ombytninger, og hvis det er lovligt og den bedste ombytning, så ender eksemplet i situation 3. Situation 3 illustrerer den løsning der returneres i linie 17.

I ovenstående pseudo-kode og illustration heraf (Figur 11 og Figur 12) er begge 1 til 1 ombytning foretaget per ordre i den udvalgte rute. Som et alternativ til dette kan man undersøge alle mulige ombytninger for hver ordre i ruten og kun foretage den bedste ombytning for hele ruten. Det svarer til, at linie 13 til 16 i pseudo-koden flyttes helt ud til venstre, så linie 13 er på niveau med linie 1 og 17. Figur 13 viser 1 til 1 ombytning per rute. Først foretages alle mulige sammenligninger, hvorefter den bedste udvælges og ombytningen foretages, såfremt den ikke er tabu. I nedenstående figur er ombytningen O_2 og O_4 bedst og situation 2 viser hvad koden returnerer.



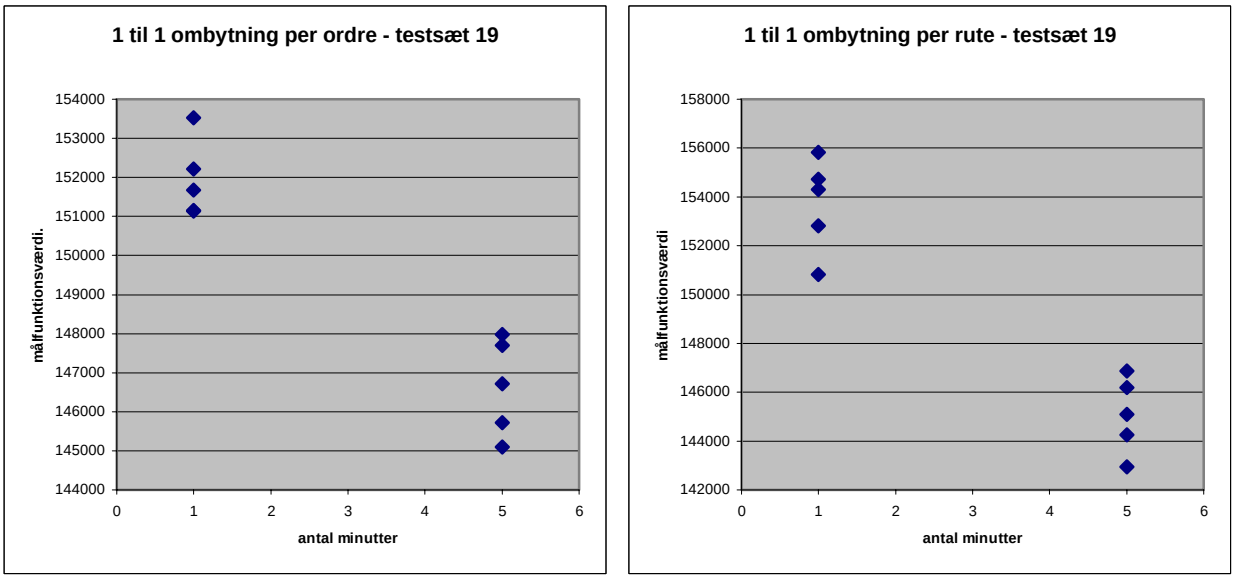
Figur 13 – 1 til 1 ombytning per rute

Fordelene ved 1 til 1 ombytning per rute, i forhold til 1 til 1 ombytning per ordre, er at nabolaget bliver større og derved er der større sandsynlighed for en god ombytning. Ulemperne er at der samlet foretages færre ombytninger i forhold til 1 til 1 ombytning per ordre. Nedenstående grafer viser forskellen på de to mht. målfunktionsværdi.



Figur 14 a og b – 1 til 1 ombytning per ordre og per rute for testsæt 7

For testsæt 7 på ovenstående figur giver 1 til 1 ombytning per rute et klart bedre resultat end 1 til 1 ombytning per ordre.



Figur 15 a og b – 1 til 1 ombytning per ordre og per rute for testsæt 19

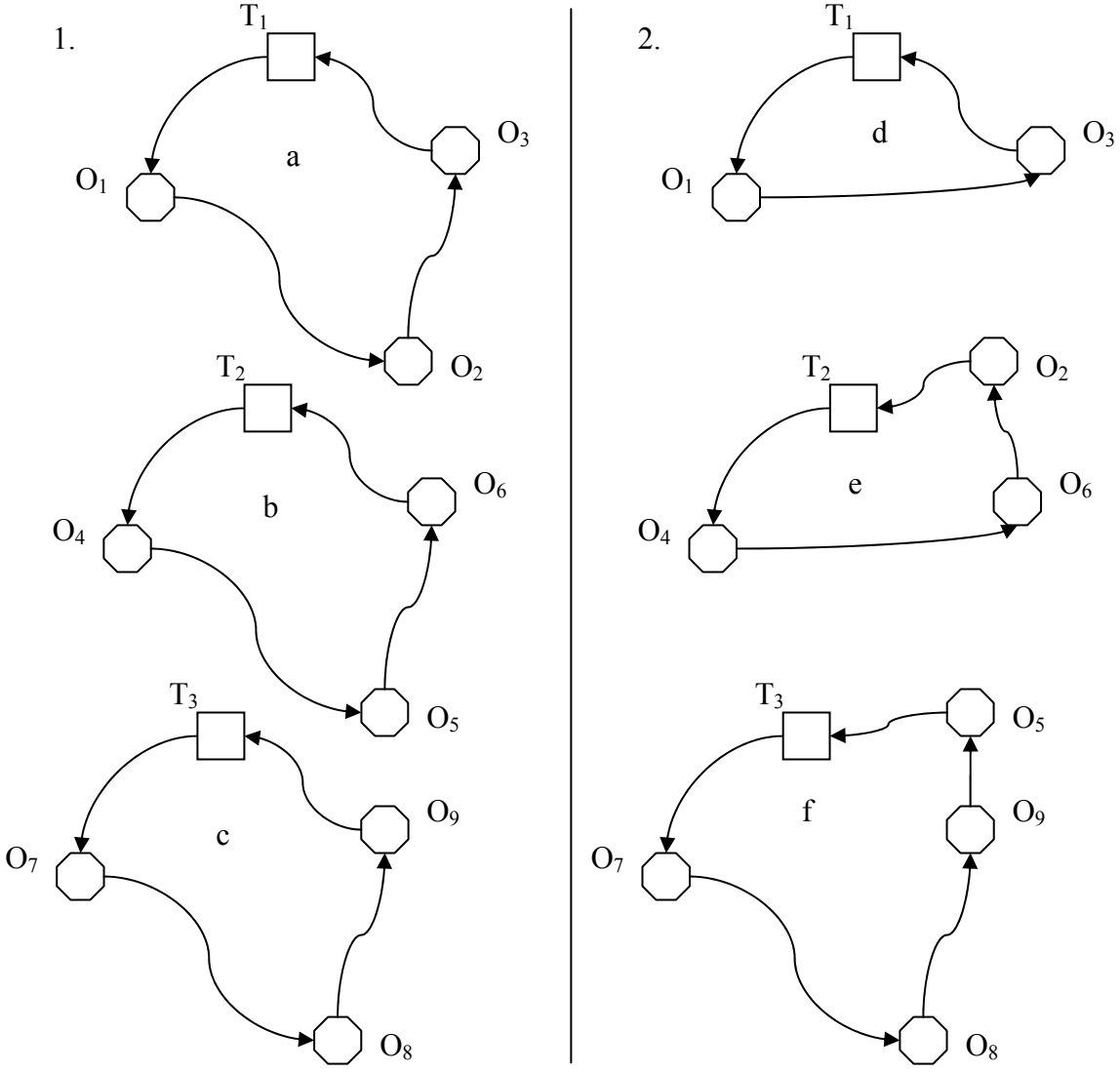
For testsæt 19 er forskellen mellem ”per ordre” og ”per rute” mindre, men 1 til 1 ombytning per rute har lidt bedre resultater. Samme tendens har jeg set i andre tests. Derfor har jeg valgt at benytte 1 til 1 ombytning per rute for fremtiden.

5.3.2 1 til 0 ombytning

For at komme væk fra den fastlåste situation i 5.3.1, hvor antallet af knudepunkter per tur og rute er fastlagt, konstruerede jeg en 1 til 0 ombytning. Derved er det muligt at ændre strukturen fra den udspecificeret i den initielle løsning. Men der var flere negative konsekvenser ved 1 til 0

ombytning, hvoraf den ene er beskrevet som en fordel ved 1 til 1 ombytning, nemlig evnen til at bytte ordrer mellem allerede fyldte ture. Problemet kommer i 1 til 0 ombytning hvis en ordrer forsøges flyttet, men mange af de attraktive løsninger er blokeret fordi turene allerede er fyldte. Nabolaget bliver mindre end for en 1 til 1 ombytning og det samme gør sandsynligheden for at finde en god ombytning. Det sidste skyldes, at der ikke er grund til at tro, at hver enkelt mulig ombytning i 1 til 0 ombytning er bedre end hver enkelt mulig ombytning i 1 til 1 ombytning. Den eneste forskel er størrelsen af nabolaget og eksperimentelt er det betydeligt mindre ved 1 til 0 ombytning end ved 1 til 1 ombytning.

En mulig løsning til ovenstående problem er ejection-chains. Ejection-chains virker ved at flytte et kapacitetsproblem videre, indtil en lovlig løsning opnås. Figur 16 illustrerer dette. I situation 1 skal ordre O_2 flyttes fra a til b. Desværre har b ikke kapacitet til O_2. Foretages flytningen alligevel bliver b ulovlig, dog kan det repareres ved at flytte en ordre, der bringer kapaciteten ned på et acceptabelt niveau, videre til en anden rute. Flyttes O_5 til c, hvor der er nok kapacitet, er problemet løst.



Figur 16 – Ejection-chains til 1 til 0 ombytning

Der var desværre ikke tid til at eksperimentere med ejection-chains i dette projekt, men det er en udvidelse der synes at være potentiale i.

5.3.3 1 til 2 ombytning

Som beskrevet i [Røp02] kan det være en god ide, at tage en serie af ordrer ud fra en tur og indsætte dem i en anden tur. Ideen er, at nogle af ordrerne i en tur ligger i nærheden af hinanden, så det er en fordel at beholde dem samlet. Fra den initielle løsning må man også forvente, at ordrer i samme tur ligger forholdsvis tæt på hinanden, da opbygningen af den initielle løsning sker via en grådig algoritme, der forsøger at minimere kørselstiden. Dette gælder nødvendigvis ikke hele tiden, men som antallet af iterationer stige,r må man forvente at løsningerne bliver bedre og dermed at ordrerne i hver tur kommer til at ligge godt i forhold til hinanden. Jeg har valgt at sætte længden af en serie til 2. For at undgå ulemperne ved ikke at bytte med en anden ordrer, kombineres serien af 2 ordrer med 1 ordrer fra en anden rute. Dette giver en 1 til 2 ombytning. Som for 1 til 1 ombytning foregår 1 til 2 ombytningen kun en gang for en rute per iteration. Herunder er vist pseudo-koden for 1 til 2 ombytning virkende med en ombytning per rute. Den største forskel i pseudo-koden fra 1 til 1 ombytning er i linie 5, 7 og 8 hvor der nu er henholdsvis geografisk afgrænsning og behandles to knudepunkter i stedet for et.

```
1   Vælg tilfældig rute a
2        For hver tur
3             For hver knudepunkt
4                  Hvis knudepunkt er ordre
5                       For hver rute (minus rute a) hvis kasse overlapper rute a's kasse
6                            For hver tur
7                                 For hvert sæt af 2 knudepunkter
8                                      Hvis knudepunkterne er ordrer
9                                           Foretag ombytning
10                                          Kontroller om løsningen er lovlig
11                                               Hvis ja så gem løsningen
12                                               Hvis nej så ombyt til oprindelig løsning
13  Vælg ombytningen svarende til bedste gemte løsning
14       Kontroller om ombytningen er tabu
15            Hvis ja så slet ombytningen fra de gemte løsninger og spring til 13
16            Hvis nej så foretag ombytningen
17  Returner løsning
```

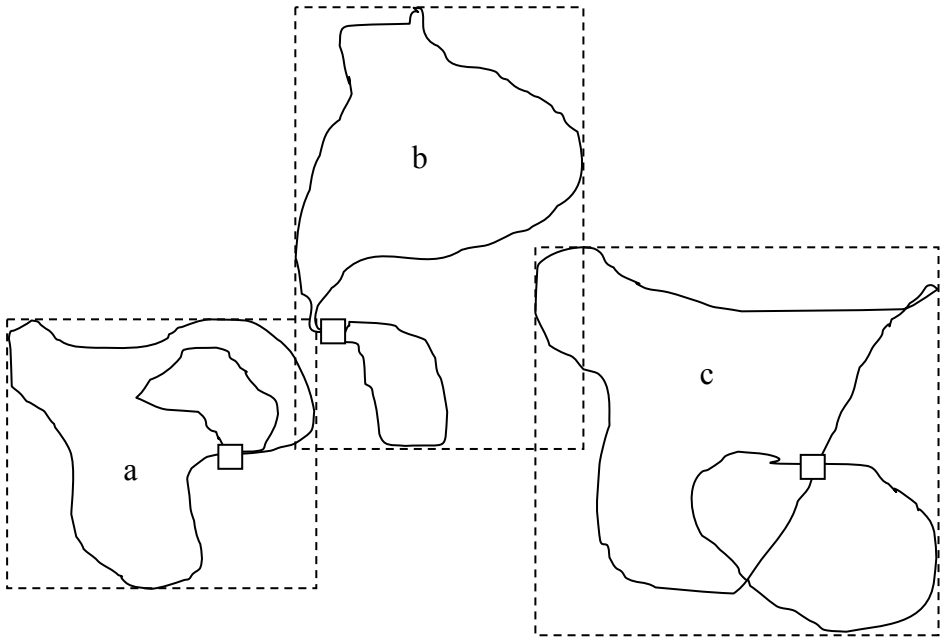
Figur 17 – Pseudo-kode for 1 til 2 ombytning per rute

Rent eksperimentelt kan jeg se, at 1 til 2 ombytning gennemløber omkring dobbelt så mange iterationer, som 1 til 1 ombytning på samme tid. En mulig forklaring på dette kunne være, at 1 til 2 ombytning giver anledning til færre mulige ombytninger per iteration end 1 til 1 ombytning. Nabolaget for 1 til 2 ombytning er simpelthen mindre end nabolaget for 1 til 1 ombytning, hvorfor 1 til 2 ombytning hurtigere kan gennemføre flere iterationer. En 1 til 2 ombytning for en hel rute tager lidt længere tid at gennemføre end en 1 til 1 ombytning for en hel rute, men en 1 til 2 ombytning skal kun prøve n-1 ombytninger per tur, hvor n er antal ordrer, mens en 1 til 1 ombytning skal prøve alle n muligheder. Er n mellem 1 og 5, hvilket er den typiske størrelse, svarer n-1 til en besparelse på mellem 20% og 100%. Desuden kontrolleres vægtbegrænsningen forholdsvis hurtigt i en ombytning, hvilket igen taler til fordel for 1 til 2 ombytning. Antager vi at det i gennemsnit er mindre sandsynligt, at 2 ordrer kan passe ind i en ny rute end 1 ordre kan rent vægtmæssigt, kan 1 til 2 ombytning hurtigt bortkaste et antal ombytninger på vægtbegrænsningen alene, hvilket sparer en del tid.

I afsnit 5.3.5 vil jeg vise en sammenligning af de ovenstående typer af ombytning inklusiv geografisk afgrænset ombytning, som jeg nu vil gennemgå. Desuden vil jeg også se nærmere på nabolagenes størrelser.

5.3.4 Geografisk afgrænset ombytning

Ideen bag geografisk afgrænset ombytning er, at frasortere usandsynlige ombytninger på forhånd, på en måde der er hurtigere end at afprøve en ombytning, hvorved en tidsbesparelse opnås. Ideen stammer fra [Røp02]. Den geografiske afgrænsning er i denne opgave valgt som et rektangel og man kan betragte det som en kasse, der er lagt ned over hver rute. Jeg vil derfor referere til rektanglet som en kasse fra nu af. Kassen udregnes på baggrund af rutens yderste grænser, dvs. x og y koordinaterne for alle knudepunkter sorteres og det mindste og største koordinat for henholdsvis x og y udgør grænserne. Vognen kan godt køre udenfor kassen, hvis ruten mellem to knudepunkter går udenfor kassen. Et eksempel på geografisk afgrænsning kan ses på Figur 18. Her er 3 ruter afbilledet, med terminalerne markeret med en lille firkant og den geografiske afgrænsning påtegnet, som en stiplet kasse. På billedet består rute a, b og c af 2 ture hver. Som man kan se, overlapper rute b med begge de andre ruter, mens ingen af disse overlapper med hinanden. Ville man ombytte en ordre fra rute a, ville rute c være udelukket og rute b den eneste mulighed i dette scenario. Det betyder ikke, at det er umuligt at betjene en ordre på rute c fra terminalen i a, men det er bare mere usandsynligt at det fører til en god løsning end en ombytning med en ordre fra rute b. Derfor sorteres muligheden fra på forhånd, hvorved algoritmen sparer tid.



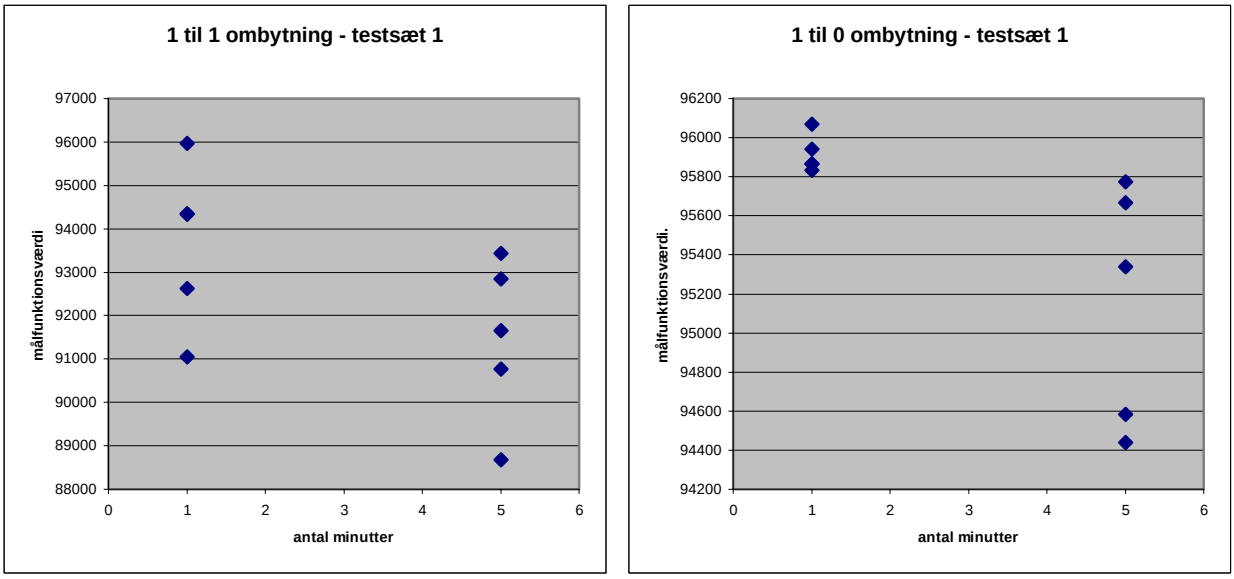
Figur 18 – Geografisk afgrænsning af ruter

Umiddelbart kunne det virke bedst, hvis kassen blev udregnet ud fra turen, da det er mellem ture man bytter ordrer. Men det ville afgrænse området uhensigtsmæssigt meget. Ligeledes kunne man have en geografisk afgrænsning, der kunne følge en rute mere præcist end en kasse kan, men selve afgrænsningen skal være hurtigere og lettere at kontrollere end faktisk at forsøge en ombytning.

Bliver den geografiske afgrænsning for kompleks eller for finmasket, mister man hastighed og tidsmæssigt begynder situationen at ligne en uden geografisk afgrænsning. Det er en fin balance at implementere en geografisk afgrænsning, så de mest usandsynlige ombytninger (rent målfunktionsmæssigt) frasorteres, mens de sandsynlige ombytninger bibeholdes, samtidigt med at antallet af afgrænsninger holdes på et acceptabelt niveau.

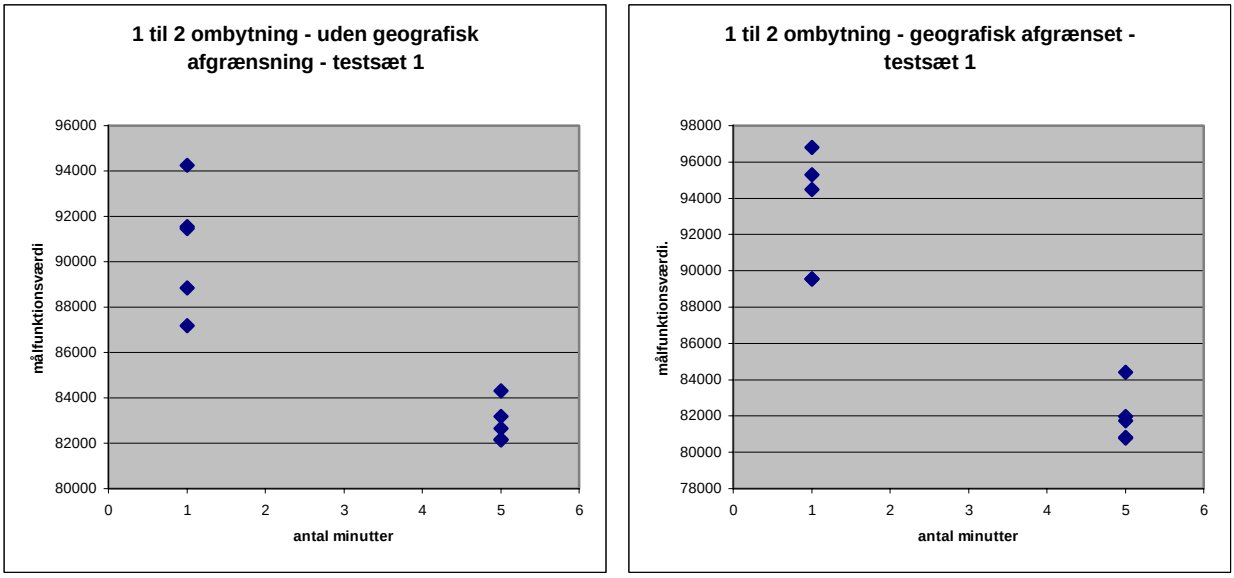
5.3.5 Sammenligning

Herunder sammenholder jeg resultaterne fra 4 forskellige testkørsler fra henholdsvis testsæt 1 og 19 (valgt tilfældigt blandt testsættene). Formålet er at kunne sammenligne hvilken type ombytning, der giver den bedste målfunktionsværdi. Til dette er der foretaget 5 kørsler af 1 minuts varighed og 5 kørsler af 5 minutters varighed. Først ser vi på testsæt 1:



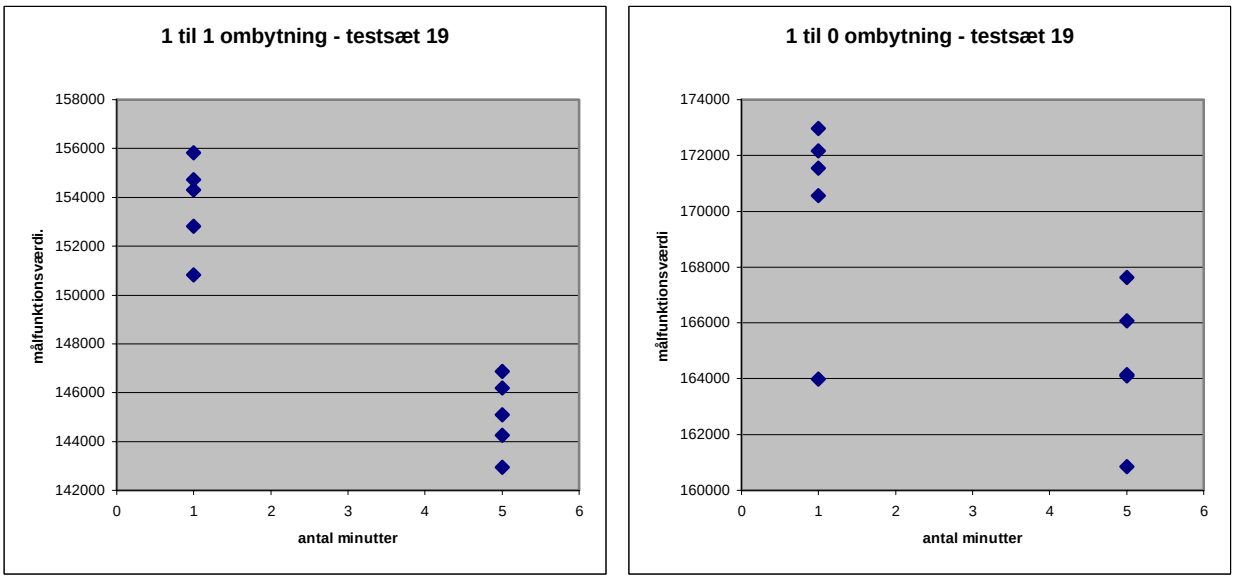
Figur 19 a og b – resultater fra 1 til 1 og 1 til 0 ombytning for testsæt 1

Testsæt 1 på Figur 19a viser tydeligt at 1 til 1 ombytning er bedre end 1 til 0 ombytning på Figur 19b, især ved 5 minutters kørselstid.



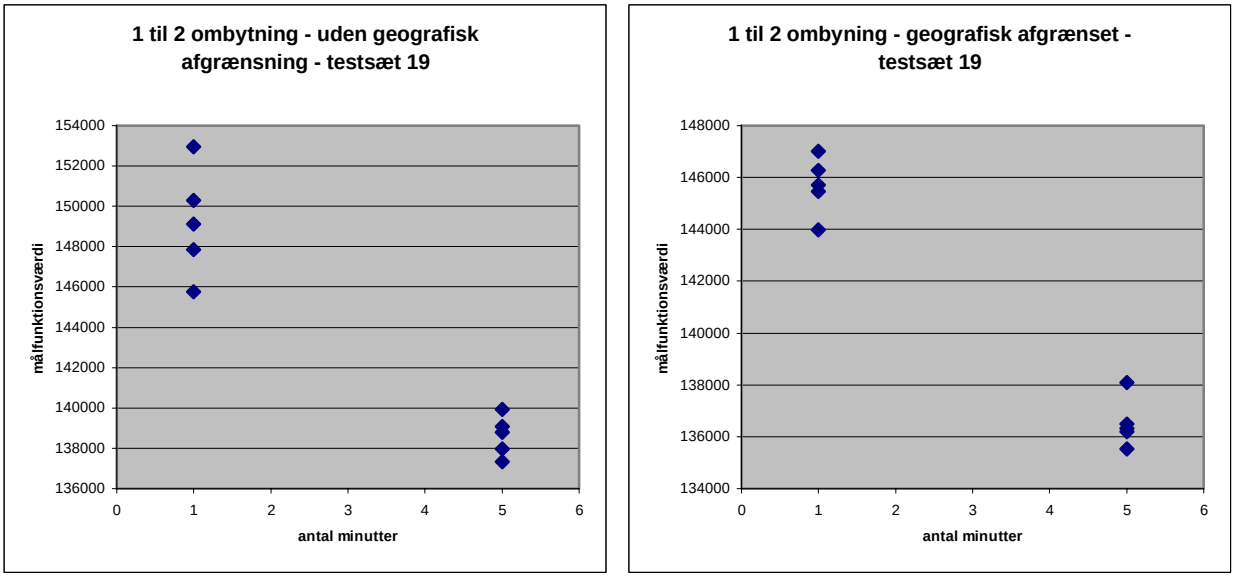
Figur 20 a og b – resultater fra 1 til 2 ombytning uden og med geografisk afgrænsning for testsæt 1

På Figur 20a er resultaterne fra 1 til 2 ombytning uden geografisk afgrænsning vist. På Figur 20b er geografisk afgrænsning desuden aktiveret og generelt giver det et bedre resultat. Sammenlignes Figur 19 og Figur 20 er det tydeligt, at resultaterne fra Figur 20 er overlegne i forhold til Figur 19. 1 til 2 ombytning med geografisk afgræsning er at foretrække for testsæt 1. Nedenunder er samme analyse foretaget for testsæt 19:



Figur 21 a og b – resultater fra 1 til 1 og 1 til 0 ombytning for testsæt 19

Figur 21 giver anledning til samme konklusioner som Figur 19. 1 til 1 ombytning er overlegen i forhold til 1 til 0 ombytning mht. målfunktionsværdien efter 1 og 5 minutters kørsel.



Figur 22 a og b – resultater fra 1 til 2 ombytning uden og med geografisk afgrænsning for testsæt 19

Igen giver testsæt 19 på Figur 22 tilsvarende konklusioner som testsæt 1 på Figur 20. Begge former af 1 til 2 ombytning er overlegne i forhold til 1 til 1 og 1 til 0 ombytning. Set over begge testsæt er 1 til 2 ombytning med geografisk afgræsning at foretrække.

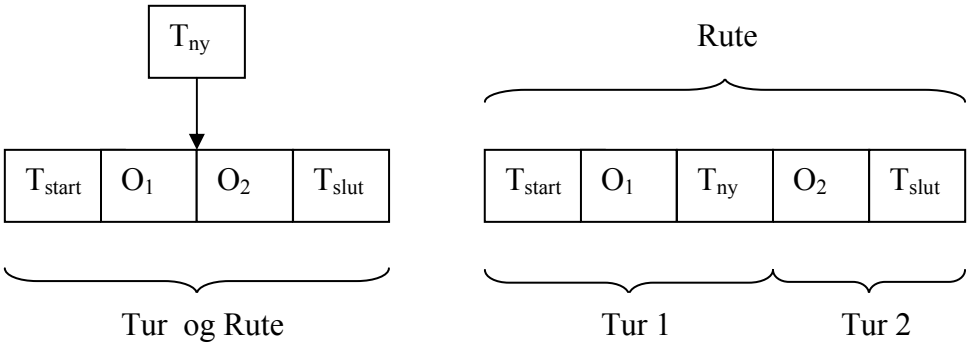
Nabolagets størrelse er også interessant. Det er ikke let at sammenligne nabolagets størrelse for en iteration mellem kørsler ”per ordre” og ”per rute”. Da jeg har valgt at køre ”per rute”, vil jeg derfor kun sammenligne disse nabolags størrelse. Nabolagets størrelse i gennemsnit over 5000 iterationer er for 1 til 1 ombytning 76,9 og for 1 til 2 ombytning 34,6. Denne opførsel har jeg tidligere omtalt, da 1 til 1 ombytning har flere mulige ombytninger end 1 til 2 ombytning og en højere sandsynlighed for at overholde vægtbegrænsningerne for vognene. Tilføjes geografisk afgrænsning til 1 til 2 ombytning falder nabolagets størrelse til 28,6. Faldet i nabolagets størrelse ved indførelse af geografisk afgrænsning er forventet. Faldet er dog forholdsvist lille, hvilket signalerer at mange ruter overlapper hinanden.

I de følgende afsnit vil jeg gennemgå yderligere hjælpefunktioner til at forbedre de ovenstående typer af ombytninger.

5.3.6 Indsættelse af terminal

I alle ovenstående typer af nabolag ligger antallet af ture per rute fast. Der kan udelukkende fjernes eller tilføjes ordrer. Det er ikke tilstrækkeligt til at garantere en god løsning, da mange mulige løsninger stadigt ikke kan nås. Derfor er det nødvendigt at kunne indsætte nye terminaler. Dette gør jeg ved at indsætte en terminal mellem to ordrer i en rute. Dette gentages for alle eksisterende terminaler og alle par af ordrer i ruten. Herefter sorteres indsættelserne efter målfunktionsværdi og den bedste indsættelse kan nu vælges.

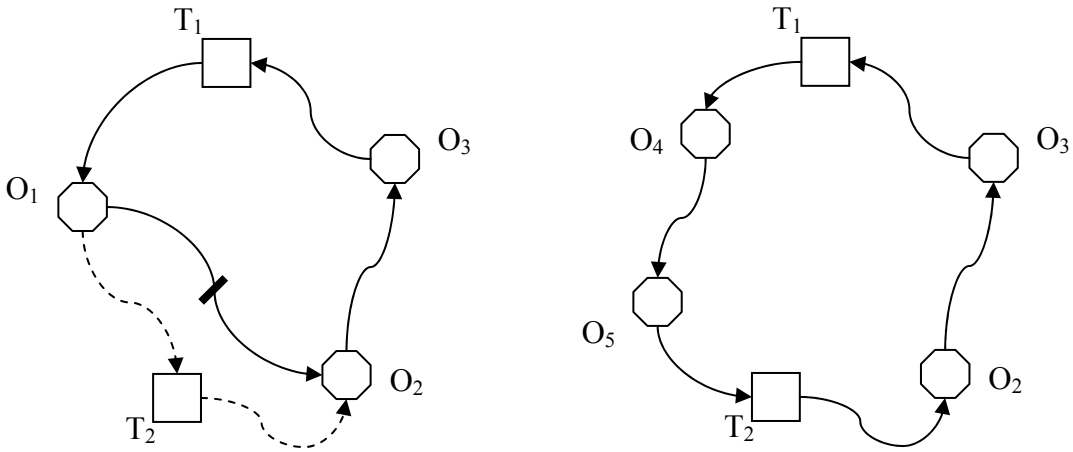
Selve indsættelsen af en ny terminal medfører oprettelsen af en ny tur i ruten, som illustreret på nedenstående figur.



Figur 23 – Indsættelse af terminal øger antallet af ture med 1

Kørselsmæssigt vil selv den bedste af de mulige terminaler give en løsning, der har dårligere målfunktionsværdi end den eksisterende løsning, da hver vogn forsøger at køre den korteste rute mellem de punkter den skal besøge. Et ekstra element i en korteste sti vil aldrig give et bedre resultat, pga. trekantsuligheden. Der er også produktionsomkostninger at tage hensyn til, så nogle indsættelse kan give et positivt resultat samlet, eftersom ordrerne på den nye tur mister deres gamle produktionssted til fordel for det nye (der kan være billigere). På ovenstående figur svarer det til, at ordre O_2 før blev serviceret med en produktionsmængde fra terminal T_{start}, men efter indsættelsen bliver serviceret med en produktionsmængde fra T_{ny}. Terminalen med den laveste produktionsomkostninger kan aldrig få en bedre løsning ved kun at indsætte en terminal, men det er for kompliceret at indsætte både en terminal plus et antal ordrer på en gang. Det er derfor nødvendigt at acceptere, at en indsættelse af en terminal typisk giver en forværring af målfunktionsværdien. Tillader jeg at alle terminaler kan optræde som T_{ny}, vil T_{ny} i langt de flest tilfælde være samme terminal som T_{start} og T_{slut}. Kun i det tilfælde hvor en tur eller rute går meget tæt forbi en anden terminal, der har lavere produktionsomkostninger, kan denne anden terminal indsættes som T_{ny}. Indsættelse af samme terminal som ruten starter og slutter ved er ikke særlig interessant. Som følge heraf har jeg besluttet, at der kun må indsættes terminaler, der ikke er identisk med start- og slutterminalen for ruten.

Figur 24 illustrerer indsættelsen af en terminal, der ikke er identisk med start- og slutterminalen. Umiddelbart bliver den samlede rute længere pga. trekantsuligheden, der bliver to ture og produktionsmængden til at servicere ordre O_2 og O_3 henholdsvis fjernes fra terminal T_1 og tilføjes til T_2. Efter bare en iteration af 1 til 2 ombytning kan situationen se ud som på næste billede hvor O_1 er erstattet af O_4 og O_5. Denne situation kan give en samlet forbedring, i forhold til før indsættelsen af den nye terminal, men det er svært at forudsige før man foretager indsættelsen af terminalen plus de følgende ombytninger.



Figur 24 – Indsættelse af en terminal

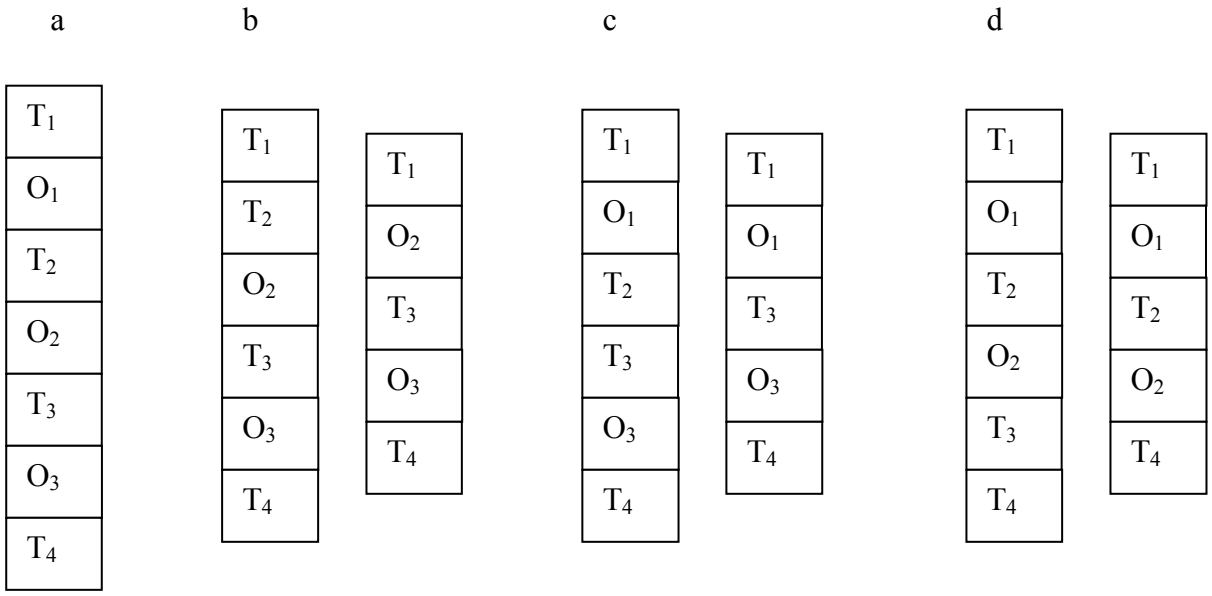
Da man ikke kan garantere, at indsættelsen af en terminal i sidste ende giver en forbedret målfunktionsværdi, er det et problem hvis terminalen ikke kan fjernes igen. At fjerne en terminal fra en rute kunne gøres ligesom indsættelsen af en terminal. Problemet ved denne fremgangsmåde er, at beslutte hvornår en terminal skal fjernes og om det kan lade sig gøre. Der er ingen garanti for, at man kan fjerne terminal T_2 på ovenstående figur uden at overskride vognens lasteevne eller produktionskapaciteten for T_1. Jeg har valgt en anden metode, der kun fjerner en terminal fra en tur, hvis terminalen er ubenyttede mht. opsamling af last. Dette indebærer, at alle ordrer terminalen betjener på den givne tur er flyttet til andre ture eller ruter. Den opmærksomme læser vil bemærke, at 1 til 1 ombytning og 1 til 2 ombytning aldrig kan flytte den sidste ordre på en tur. Derfor skal en funktion til at flytte den sidste ordre på en tur, kombineres med funktionen til at fjerne en terminal. Samlet kalder jeg det en konsolideringsfunktion.

5.3.7 Konsolideringsfunktion

Som beskrevet i tidligere afsnit foretages en række ombytninger af ordrer og indsættelser af terminaler under algoritmens kørsel. Fælles for 1 til 1 ombytning og 1 til 2 ombytning gælder at en tur eller rute ikke kan blive tom, hverken generelt eller mht. ordrer. Det betyder at hverken ture, ruter eller terminaler kan fjernes. Som tidligere beskrevet kan indsættelse af terminaler fører til nogle ture og ruter der er dårlige, selv efter et antal ombytninger er foretaget. Derfor skal konsolideringsfunktionen kunne flytte den sidste ordre fra en tur, kunne fjerne ubenyttede terminaler og kunne nedlægge ubenyttede ture og ruter.

For det første prøver den at flytte alle singleordrer over i andre ture, så den nu ubenyttede terminal kan fjernes. Med singleordrer menes ordrer der er alene i en tur, som illustreret under a på Figur 25. Her har vi 4 terminaler, der alle godt kan være den samme terminal, og 3 ordrer, der alle er singleordrer. Ruten består af 3 ture og T_1 er identisk med T_4 og ingen af disse må fjernes. Skal en af singleordrerne fjernes, ender man i en af de tre situationer illustreret som b, c eller d på figuren. I b er ordre 1 fjernet og to terminaler kommer nu efter hinanden. Hvis T_1 og T_2 ikke er samme terminal kan ingen af dem fjernes, da vognen skal starte i T_1 og produktionsmængden kommer fra T_2. Denne situation svarer til at vognen kører direkte fra terminal 1, uden at medtage last til terminal 2, hvorfra lasten til ordre 2 bliver opsamlet. Antallet af ture er uændret, da ingen terminal er fjernet. Hvis T_1

og T_2 er samme terminal, kan den ene af disse fjernes, hvorved en tur også fjernes. Man har nu den anden situation under b. I c er ordre 2 fjernet og det er nu formålsløst at køre til terminal 2, da der ikke må opsamles last der. Terminal 2 fjernes sammen med sin tomme tur og vognen kører direkte fra ordre 1 til terminal 3. I d er ordre 3 fjernet og situationen er tæt på den samme som i c. Der er ingen grund til at køre til terminal 3, hvorfor denne fjernes og man kører i stedet direkte til terminal 4, hvor ruten ender.



Figur 25 – Konsolideringsfunktioon i aktion

At fjerne en ordre fra ruten på Figur 25 medfører 4 forskellige udfald, alt efter hvilken ordre der fjernes og om T_1 og T_2 er samme terminal.

5.3.8 Oprettelse af ny rute

Da konsolideringsfunktionen kan nedlægge ture og ruter har jeg besluttet, at det skal være muligt at oprette en ny rute. Den eneste måde at oprette en ny rute, der fungerer med 1 til 1 og 1 til 2 ombytning, er at flytte en eller flere ordrer fra en eksisterende rute til den nye rute. Da projektet er i sin sidste fase, valgte jeg at gøre det på en forholdsvis simpel måde. Jeg tager en hel tur, der starter og slutter ved samme terminal, som den nye rutes vogn tilhører og flytter denne tur over som den nye rute. Eneste krav til sådan en flytning er, at den nye rutes vogn skal have en maksimal lasteevne større end turens samlede vægt. Alle andre parametre forbliver uændret. Terminalen har samme produktion og der er samme samlede kørselstid og kørselsdistance. Hvis kun en ordre eller en del af en tur var flyttet over, som den nye rute, ville terminalens produktion stadigt være uændret, men de andre faktorer ville være forskellige.

Oprettelse af en ny rute er kun muligt, hvis en rute er blevet nedlagt af konsolideringsfunktionen. Hver gang en eller flere ruter bliver nedlagt af konsolideringsfunktionen, kaldes oprettelse af ny rute og hvis det er muligt oprettes 1 ny rute. Antallet af ruter i en løsning kan ikke overstige antallet af ruter i den initielle løsning, men antallet af ruter kan godt falde, hvis det ikke var muligt at oprette en ny rute eller hvis mere end 1 rute nedlægges samtidigt.

5.3.9 Sammenfatning

Jeg har valgt at se bort fra nabolaget med 1 til 0 ombytning. Der er flere grunde til dette valg:

- Der er ikke tid til at teste alle 3 nabolag.
- 1 til 0 ombytning virker ikke lovende i forhold til de to andre ombytninger
- 1 til 1 ombytning opfatter jeg selv om en klassisk ombytning, der er god at have med som sammenligningsgrundlag, hvorfor jeg ikke vil bortkaste denne.
- 1 til 2 ombytning er det nabolag jeg har fokuseret mest på og som jeg har de største forventninger til, hvorfor jeg ikke vil bortkaste denne.

Efterfølgende arbejdes derfor kun med 1 til 1 ombytning og 1 til 2 ombytning og deres hjælpefunktioner.

I resten af rapporten vil 1 til 1 ombytning referere til 1 til 1 ombytning fra afsnit 5.3.1 inklusiv konsolideringsfunktionen fra afsnit 5.3.7, såfremt andet ikke er nævnt.

1 til 2 ombytning vil referere til 1 til 2 ombytning fra afsnit 5.3.3 med geografisk afgrænsning fra afsnit 5.3.4, indsættelse af terminal fra afsnit 5.3.6 og konsolideringsfunktionen fra afsnit 5.3.7.

Desuden vil konsolideringsfunktionen fra afsnit 5.3.7 indeholde oprettelse af ny rute fra afsnit 5.3.8.

Samlet set haves nedenstående pseudo-kode for algoritmen til vores distributions- og produktionsplanlægningsproblem. Bemærk at algoritmen benytter samme type ombytning under en kørsel, med undtagelse af hjælpefunktioner, der kaldes udover den valgte type ombytning. Der skiftes ikke mellem 1 til 1 ombytning og 1 til 2 ombytning under samme kørsel. Det er et valgt jeg har truffet, da jeg gerne vil teste de forskellige ombytninger hver for sig. Der er dog intet der udelukker, at flere forskellige typer af ombytning køres i samme kørsel eller at hver iteration bliver kørt med flere forskellige typer af ombytninger, hvorefter den bedste vælges. Det er bare et aspekt der ikke bliver undersøgt i denne rapport.

```
1   Indlæs data
2   Generer initiel løsning ud fra indlæst data
3   Vælg type af ombytning
4   Sæt værdi for hvornår NK og NT skal kaldes (antal iterationer uden global forbedring)
5   Antal iterationer siden sidste globale forbedring = 0
6   Så længe stopkriteriet ikke er møde gentag da
7        Hvis kriteriet for at kalde NK er opfyldt så kald NK
8        Hvis kriteriet for at kalde NT er opfyldt så kald NT
9        Ellers kald den valgte type af ombytning
10       Hvis den returnerede løsning er en forbedring i forhold til den nuværende løsning
11           Så gem løsning og sæt antal iterationer siden sidste globale forbedring = 0
12           Ellers øg antal iterationer siden sidste globale forbedring med 1
13  Udskriv bedste løsning
```

Figur 26 – Pseudo-kode for kørsel af algoritme

6 Programmer

For at komme fra en mængde data til et færdigt resultat, skal der laves et større program. Jeg valgte at programmere i C#, da det er det nyeste sprog fra C familien og meget let at gå til. Der er blandt andet automatisk oprydning og ingen pointere, så nogle af de klassiske fejl, man kan laver i C og C++, bliver per automatik undgået i C#. C++ er dog anvendt i et lille omfang i forbindelse med udregning af kørselsafstande og kørselstider, da disse data udregnes via en .dll fil skrevet i C++.

Desuden skal mit program laves, så outputtet er på passende form i forhold til import i Transvisions program, så en visualisering er mulig.

6.1 Transvisions program

Som en del af aftalen med Transvision, har jeg adgang til et af deres kommercielle produkt, Transvision Route Planner (TRP). Jeg har arbejdet med flere versioner af programmet, men hovedsageligt med version 2.1.1. TRP er et værktøj til at analysere og planlægge distribution i situationer med traditionel ruteplanlægning. TRP kan kun håndtere et produktionssted, og kan derfor ikke løse problemet i denne opgave. Derimod kan flere dele af TRP bruges til at visualisere og kontrollere løsningerne til vores problem.

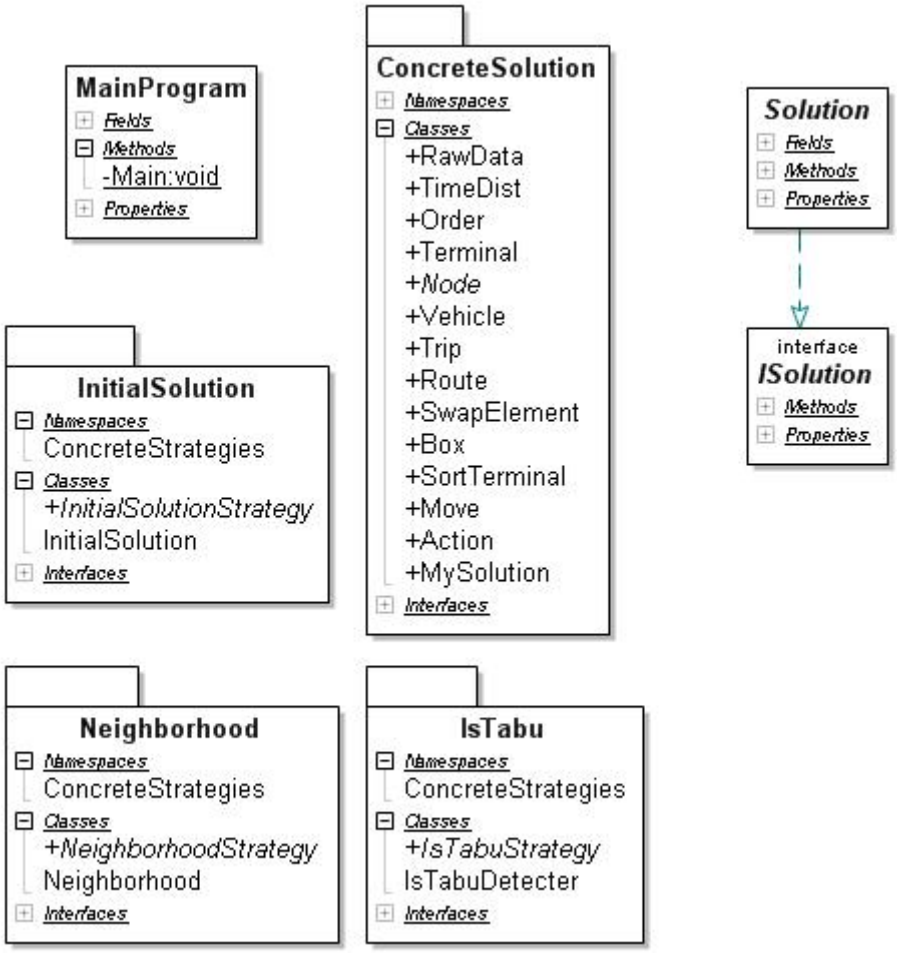
For at visualisere og kontrollere en løsning med TRP, skal løsningen leveres i et dataformat, som TRP kan importere. Desuden skal TRP konfigureres til at tolke løsningen korrekt, hvilket ikke er helt trivielt. Med hjælp fra Transvision blev begge disse problemer løst, og import af en løsning er nu rent rutinearbejde.

Importen af en løsning tager typisk flere minutter, men det afhænger fuldstændigt af størrelsen på problemet og computerkraften til rådighed.

For at begrænse udregning af de samme ting, hver gang algoritmen køres, er visse størrelser forudberegnet. Det gælder kun for kørselsafstand og kørselstid, der er de to eneste størrelser, der er forudberegnet i hele opgaven. Det er muligt at udregne disse størrelser på forhånd, da alle sæt af data er baseret på de samme 254 knudepunkter (terminaler og ordrer). To 254x254 matricer er genereret, der beskriver enhver rejse mellem 2 vilkårligt valgt knudepunkter. Hver af disse matricer tog ca. 2 timer at generere. Var udregningen af kørselsafstand og kørselstid lave dynamisk, ville det have en betydelig indvirkning på kørselstiden af algoritmen. Nu kan dette håndteres ved et tabelopslag i de to matricer. Kørselsafstanden og kørselstiden blev udregnet, ved at jeg kodede op imod en .dll fil i Transvisions program. Denne .dll fil indeholdt nogle funktioner, bla. en til at beregne den korteste kørselsafstand og mindste kørselstid mellem 2 koordinatsæt i Danmark, såfremt en hastighedsprofil var tilstede. Transvision leverede en typisk hastighedsprofil for lastbiler i Danmark og herefter blev udregningerne af kørselsafstande og kørselstider foretaget via 254 ’1 til alle’ udregninger. ’1 til alle’ udregninger er en fordel da algoritmen, der udregner kørselsafstandene og kørselstiderne, er baseret på Dijkstra’s algoritme. Dijkstra’s algoritmen finder som bekendt alle afstande fra startpunktet til ethvert besøgt punkt i en graf.

6.2 Mit program

Programmet er opbygget som illustreret på Figur 27. Fra Main kaldes alle underliggende funktioner, der indbefatter RawData, InitialSolution og Neighborhood. RawData er den del af programmet hvor data indlæses. Dataen ligger i 5 filer beskrivende terminalerne, ordrerne, vognene, distancerne mellem alle knudepunkter i problemet og tidsforbruget mellem alle knudepunkterne. Ud fra disse data genereres en initiel løsning, ved kald af InitialSolution og en trinvis forbedring sker ved gentagne kald af Neighborhood. Når det første stopkriterium opfyldes, stoppes algoritmen og programmet udskriver resultatet og anden relevant data.



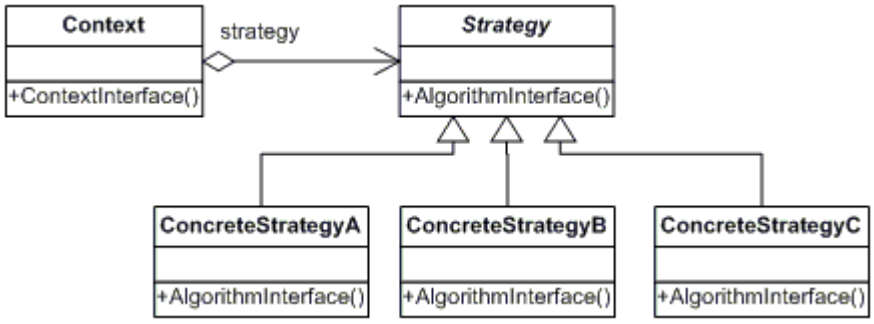
Figur 27 – Kodens struktur

Selve koden er struktureret efter to design patterns. Et design pattern er en gennemtestet designkomponent udviklet af rutinerede programmører. Det er et hjælpemiddel til andre programmører, til at strukturere et stykke kode, der skal have en vis funktionalitet, på en afprøvet og anerkendt måde. Design patterns gør det muligt at abstrahere fra detaljerne, når selve strukturen i programmet skal laves, så selv komplekse problemstillinger let kan diskuteres med udgangspunkt i det valgte design pattern.

RawData, der indlæser data fra diverse filer, er oprettet som et singleton pattern. Ideen i et singleton pattern er, at kun en instans må eksistere og denne skal kunne tilgås globalt. Hvis ingen instans eksisterer, når RawData kaldes, oprettes en ny, men hvis en instans allerede eksisterer, returneres denne blot. Derved sikres at de indlæste data ikke bliver overskrevet eller lignende. Samme kode

eller funktionalitet kunne man sagtens selv lave, men når man bruger et design pattern, er man sikker på at man ikke har overset et eller flere vigtige elementer.

InitialSolution, Neightborhood og IsTabu er alle strategy patterns. Et strategy pattern definerer en familie af algoritmer og indkapsler disse, så de kan udskiftes indbyrdes. Et eksempel på et strategy pattern kan ses på Figur 28.

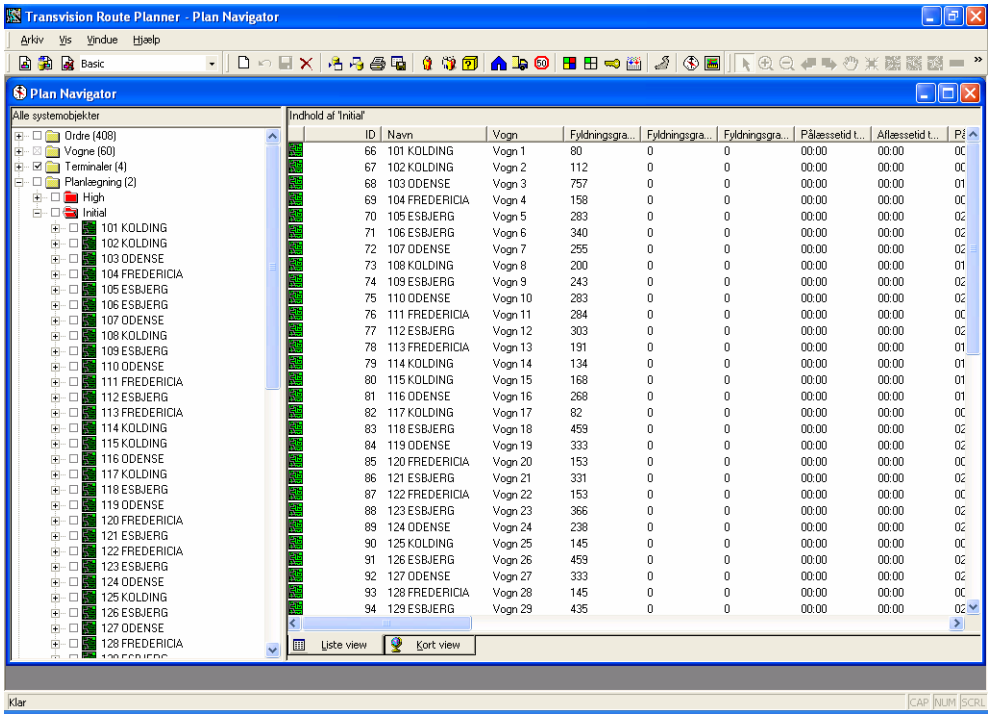


Figur 28 – Strategy pattern

Et strategy pattern muliggør en let måde at styre mange forskellige algoritmer og løbende udskifte dem, mens programmet kører. Dette udnytter jeg til at skifte mellem ombytningsalgoritmerne, konsolideringsfunktionen og indsættelse af ny terminal mens programmet kører.

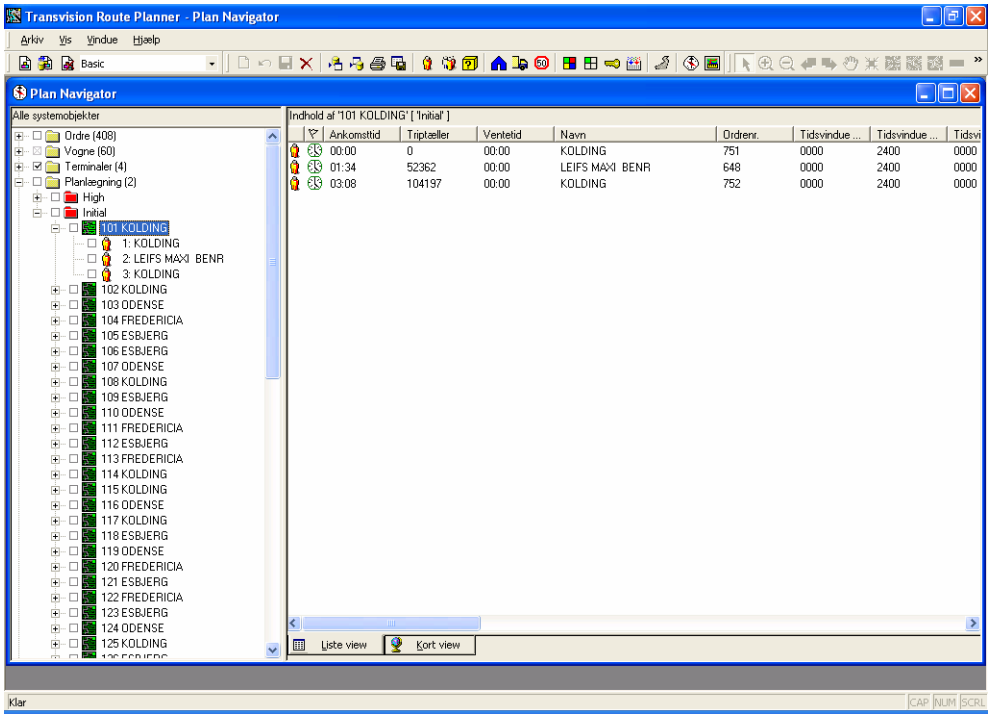
6.3 Visualisering af løsning

Efter en løsning er importeret eller genereret i TRP, er det muligt at se den repræsenteret på en listeform eller som grafik på et kort. Importerer jeg den initielle løsning, vil TRP repræsentere det som følgende på listeform.



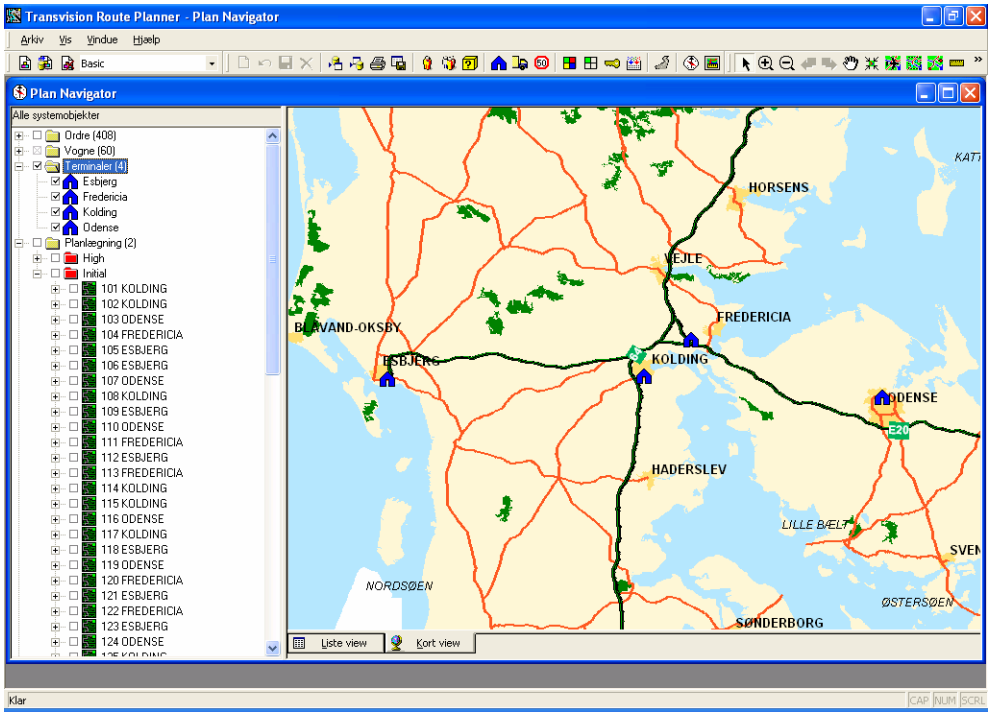
Figur 29 – En løsning på listeform

Her er de 60 ruter listet og under hver rute er det muligt at se den tilhørende vogn, dennes kørselsplan og ordrer mm. som vist på Figur 30.



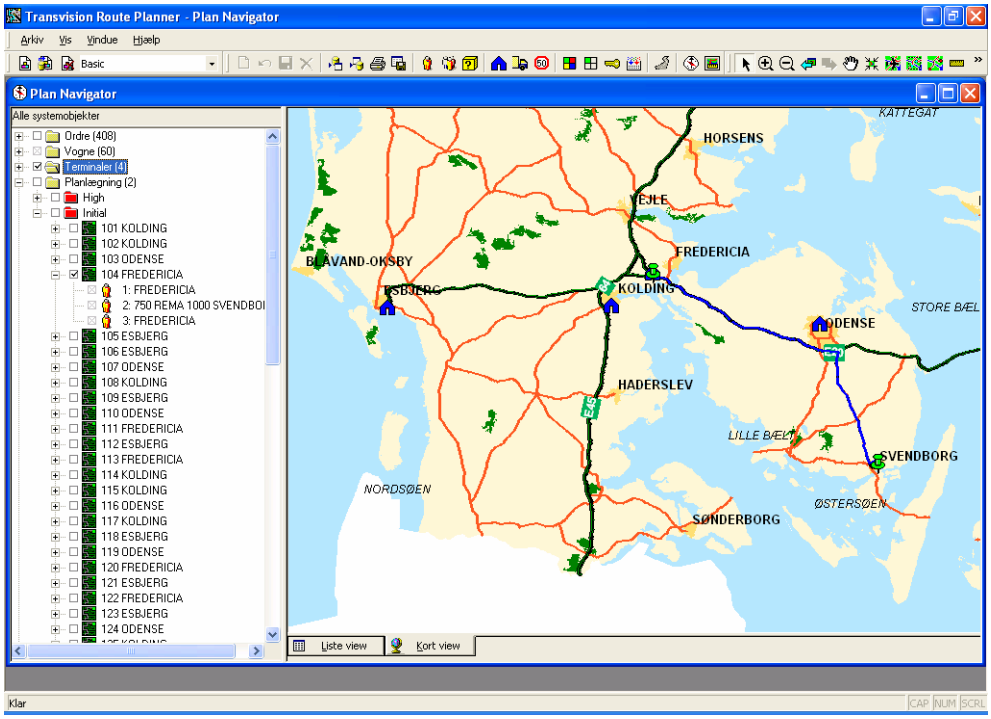
Figur 30 – En rute illustreret i TRP

Skifter vi over til visualisering på kortet ses billedet på nedenstående figur. Bemærk at der er et mærke udfor 'Terminal' hvilket indikeret at terminalerne tegnes på kortet.



Figur 31 – Kortet i TRP med terminaler markeret og påtegnet

På Figur 31 er de 4 terminaler markeret som blå huse og det øverste niveau af vejnettet er også synligt. Vælger vi at repræsentere en rute på kortet, ved at markere ruten, zoomes der automatisk ind på kortet, så ruten er det centrale element og vejnettet bliver mere detaljeret.



Figur 32 – En rute illustreret på kortet i TRP

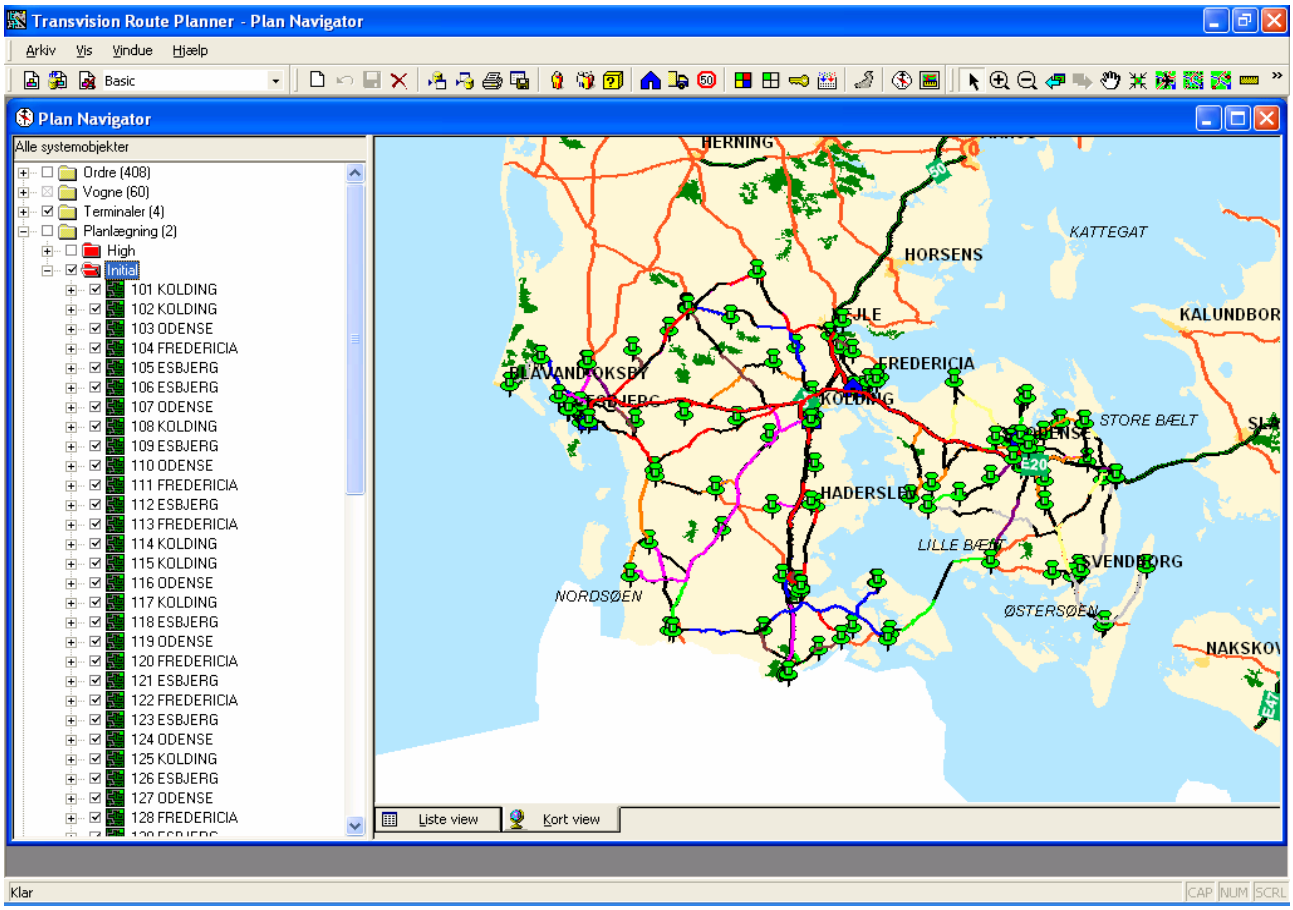
Den med blåt markerede rute er fra den initielle løsning. Ruten starter i Fredericia, kører til Svendborg (vist med et knappenålshoved) for at betjene en kunde og derefter tilbage til Fredericia. Umiddelbart virker Odense som et bedre valg til at betjene kunder i Svendborg, men i den initielle løsning er målfunktionsværdien ikke vigtig i forhold til at opnå en lovlig løsning. Ruter der er meget dårlige, vil hurtigt blive ændret, da der er en stor forbedring at hente der.

7 Tuning af algoritme og praktiske eksperimenter

I dette afsnit vil jeg beskrive hvordan algoritmen afprøves, hvordan alle parametre indstilles og hvilke datasæt der benyttes.

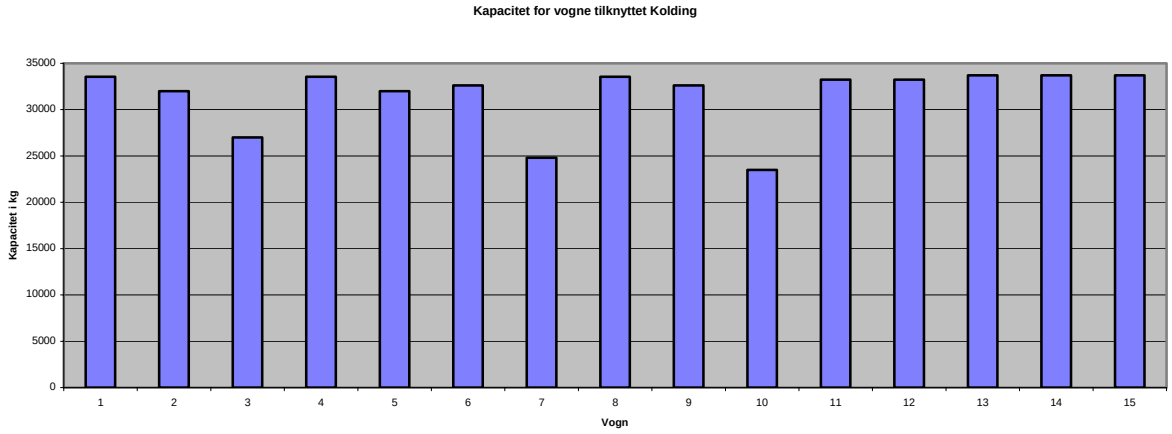
7.1 Beskrivelse af data

Med problemformuleringen blev et datasæt udleveret af Transvision. Datasættet indeholder 4 terminaler, 60 vogne og 250 ordrer. Hver af disse elementer har mange variable tilknyttet, så det fulde datasæt er ikke listet her, men kan findes i Appendiks 10.2. Terminalerne ligger fra vest mod øst i Esbjerg, Kolding, Fredericia og Odense. Ordrerne ligger hovedsageligt i det sydlige Jylland, på Fyn og Langeland som vist på Figur 33.



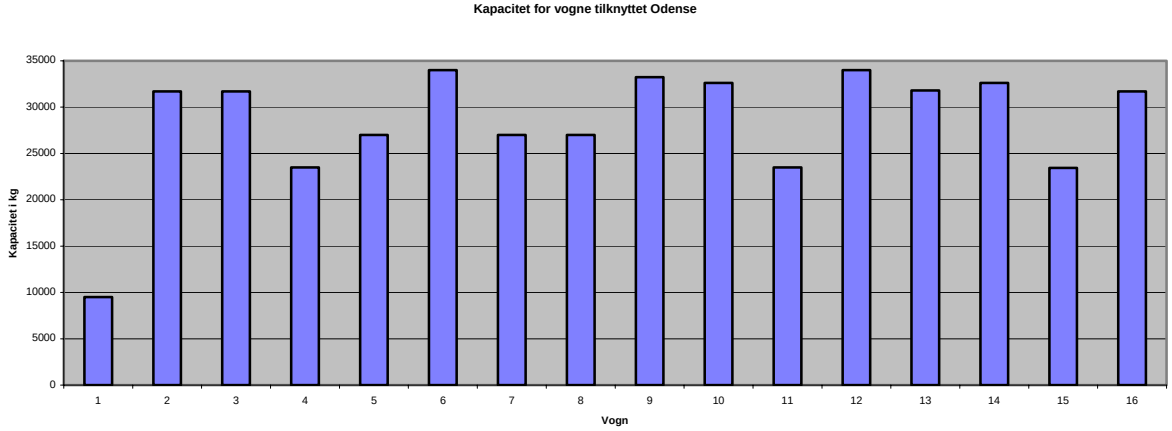
Figur 33 – Ordrernes fordeling i Danmark (bemærk at en knappenål kan repræsentere flere ordrer)

Herunder er datasættet repræsenteres grafisk på Figur 34 til Figur 37 vha. de 4 terminaler og de vogne, hver terminal har til rådighed. Figur 38 og Figur 39 viser henholdsvis de 4 terminalers data og efterspørgslen fra de 250 ordrer.



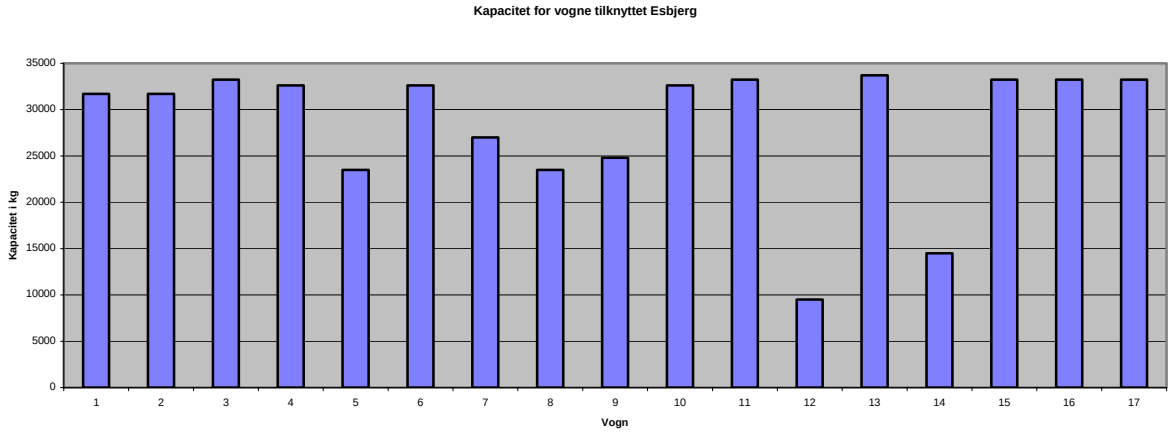
Figur 34 – Kolding

Kolding er kendetegnet ved at have et stort antal vogne i forhold til produktionskapaciteten. Kolding besidder 15 vogne, hvoraf de 12 har over 30.000 kg i kapacitet. Koldings produktionskapacitet er 400.000 til 511.000 med en pris på 0.12 kr/kg.



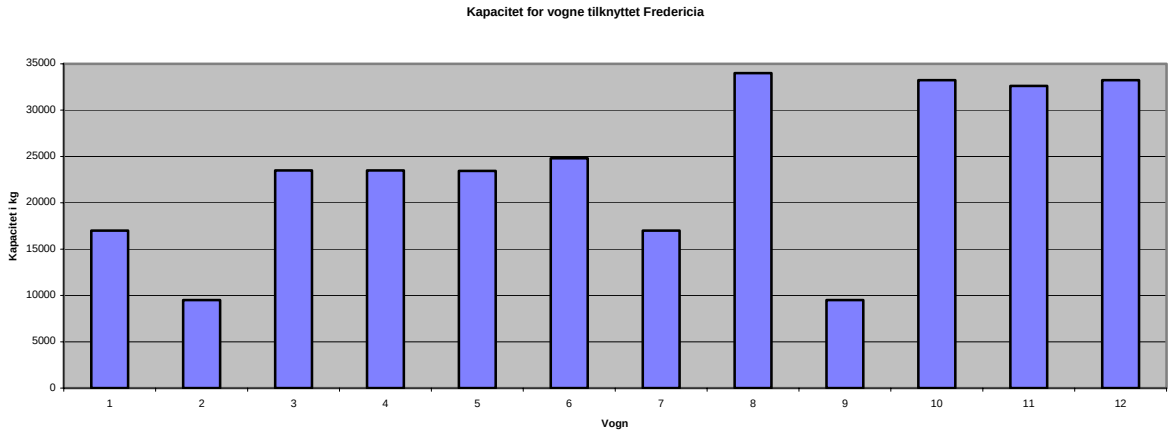
Figur 35 – Odense

Odense er den næststørste terminal med en produktionskapacitet på 600.000 til 750.000 og med en pris på 0.17 kr/kg. Odense er dermed den dyreste terminal mht. produktionspris. Odense besidder 16 vogne, hvoraf 9 er over 30.000 kg i kapacitet.



Figur 36 – Esbjerg

Esbjerg er kendetegnet ved at være den største og billigste terminal. Esbjerg råder over 17 vogne, hvoraf de 11 er over 30.000 kg i kapacitet. Produktionskapaciteten er 1.050.000 til 1.400.000 med en pris på 0.11 kr/kg.

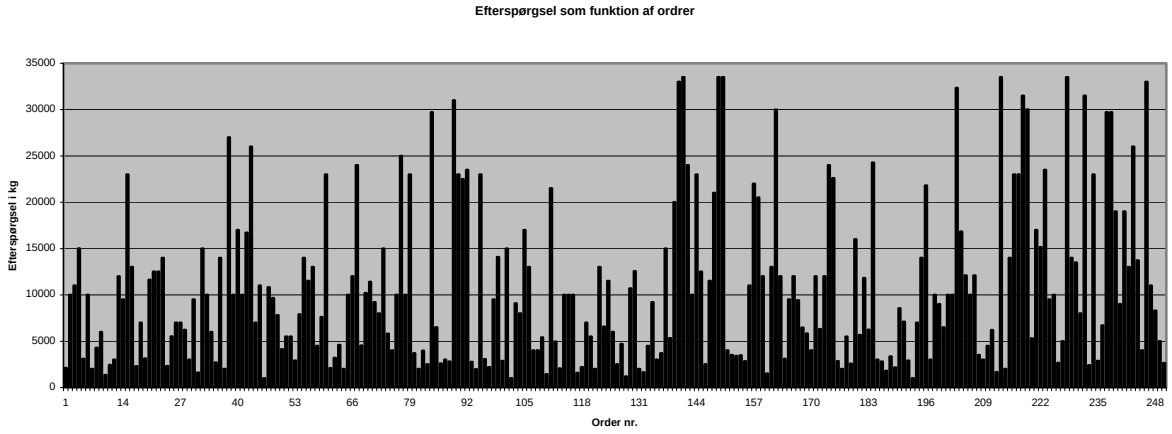


Figur 37 – Fredericia

Fredericia er helt klart den mindste terminal, både hvad angår antal vogne og produktionskapacitet. Fredericia råder over 12 vogne, hvoraf de kun 4 er over 30.000 kg i kapacitet. Produktionskapaciteten er 135.000 til 205.000 med en pris på 0.14 kr/kg.

Terminal	Min. Produktion, kg	Max. Produktion, kg	Produktions-pris, kr/kg	Antal Vogne	Antal Vogne over 30.000kg	
1 Kolding	400000	511000	0.12	15	12	80%
2 Odense	600000	750000	0.17	16	9	56%
3 Esbjerg	1050000	1400000	0.11	17	11	65%
4 Fredericia	135000	205000	0.14	12	4	33%
I alt	2185000	2866000	-	60	36	60%

Figur 38 – Opsummering af de 4 terminalers data



Figur 39 – Illustration af ordrerne

Herover ses de 250 ordrer plottet med deres efterspørgsel. En stor del af ordrerne ligger under 15.000 kg, men der er også 19 ordrer på eller over 30.000 kg.

7.2 Beskrivelse af parametre

Der er mange parametre i den foreslået algoritmen og det er ikke muligt indenfor denne opgave, at teste alle kombinationer af parametre. Derfor der må træffes nogle valg på baggrund af generelle observationer.

De vigtigste parametre for algoritmen er følgende:

1. Valg af ombytning – Det centrale punkt i algoritmen er hvilken type ombytning der vælges. Der er 2 mulige ombytninger i denne opgave og kun en af disse benyttes under en kørsel. Det er henholdsvis 1 til 1 ombytning og 1 til 2 ombytning.
2. Kald af hjælpefunktioner - Til både 1 til 1 ombytning og 1 til 2 ombytning er der flere forskellige hjælpefunktioner. Det er meget afgørende hvor ofte man kalder indsættelse af terminal eller konsolideringsfunktionen. Dette vil jeg behandle nærmere senere i rapporten.
3. Tabulistens længde - Jeg vil teste for en længden af tabulisten spændende mellem 0 og 35 i skridt af 7, plus et punkt ved 49, 60 og 120.

Overordnet kørsel af algoritmen:

- Stopkriteriet for algoritmen - Antal iterationer, antal iterationer uden forbedret løsning, samlet tidsforbrug eller opfyldt målfunktionsværdi. Jeg har valgt at stopkriteriet, der benyttes til eksperimenterne, skal være samlet tidsforbrug med værdierne 1 minut, 5 minutter, 15 minutter og 60 minutter. Disse tidsrum svarer til hvad jeg vurderer er typiske kørselstider i en anvendelsessituation. Afhængigt af eksperimentets art vil kun enkelte af tiderne benyttes, men i den endelige afprøvning vil alle 4 tider bliver gennemkørt 5 gange.

Stopkriterier	
	Antal iterationer Antal iterationer uden (global) forbedring Tidsforbrug Målfunktionsværdi

Nabolag

- Geografisk ombytning – geografisk ombytning er en parameter, der kan slås til og fra. Hvis geografisk ombytning er slået til, er det nødvendigt at de 2 valgte ruters kasser overlapper.

Problemspecifikke

- Balancering af produktionsomkostninger og kørselsomkostninger – Fra Transvisions side er det blevet forslået, at der er balance mellem de to omkostninger, da det er den mest realistiske situation. Derfor er produktionsomkostningerne skaleret med en faktor 0.28 for at opnå ca. 50/50 deling mellem produktions- og kørselsomkostninger.

Jeg vil kun beskæftige mig med de 3 vigtigste parametre i det følgende, da de andre parametre allerede nu er fastlagt. I næste afsnit vil jeg derfor tune de 3 vigtigste parametre ud fra omfattende praktiske eksperimenter.

7.3 Praktiske eksperimenter

Alle eksperimenter er udført på en Athlon 1333 MHz processor, der kører Windows XP. Jeg har forsøgt at sikre at de samme vilkår er til stede under hvert praktisk eksperiment, men i et styresystem som Windows XP vil der altid være processer og tråde der kører i baggrunden, som jeg ikke har kontrol over. Jeg må antage at det vil give en form for udsving i resultaterne, men jeg tester ikke direkte for at fastslå dette.

Det er blevet klart at et datasæt, som der foreligger ifølge opgaven, ikke er nok til en tilstrækkelig afprøvning af en metaheuristik. Flere vigtige elementer kan kun testes, hvis der anvendes et bredt udsnit af de mulige datasæt, den endelige algoritme skal virke på. At kunne generere en god løsning, til et specifikt problem, giver ingen indikation af om algoritmen generelt kan finde gode løsning eller om disse er robuste. Derfor vil jeg generere omkring 20 testsæt ud fra det oprindelige datasæt. Disse testsæt vil variere i størrelsen med hensyn til antal terminaler, vogne og ordrer. 5 af disse testsæt vil blive udvalgt og herfra vil alle parametre i algoritmen blive tunet, indtil et tilfredsstillende resultat er nået. Herefter vil algoritmen blive afprøvet på de resterende 15 testsæt, uden ændringer af parametrene.

De 20 testsæt jeg har udvalgt har følgende opbygning i deres intielle løsning:

Testsæt nr.	Terminal	Vogne	Ordrer
0	1+2+3+4	60	250
1*	1+2	31	137
2	1+2	31	119
3	3+4	26 -3	151
4*	3+4	26 -3	128
5	1+2+3	48	237
6	1+2+3	48	226

7*	1+2+4	40 -3	158
8	1+2+4	40 -3	136
9	1+3+4	41 -3	186
10	1+3+4	41 -3	168
11	2+3+4	42 -3	224
12	2+3+4	42 -3	208
13	1+2	26 -5	133
14*	1+2	15 -16	126
15	3+4	22 -7	141
16	3+4	17 -12	138
17	1+2+3	36 -12	230
18	1+2+4	27 -16	151
19*	1+3+4	34 -10	178
20	2+3+4	38 -7	220

Figur 40 – Testsæt

De 5 testsæt markeret med * er de udvalgte testsæt, der bliver benyttet til tuning af algoritmen. Tallene i Terminal søjlen refererer til terminalerne afbildet på Figur 41. Søjlen for Vogne indeholder antal vogne i den initielle løsning, efterfulgt af forskellen mellem antal mulige vogne og det faktiske antal. F.eks. for testsæt 4 er der 26 vogne i løsningen men der er 29 vogne til rådighed, dvs. 3 vogne bliver ikke brugt.

Terminal	By
1	KOLDING
2	ODENSE
3	ESBJERG
4	FREDERICIA

Figur 41 – Terminalerne

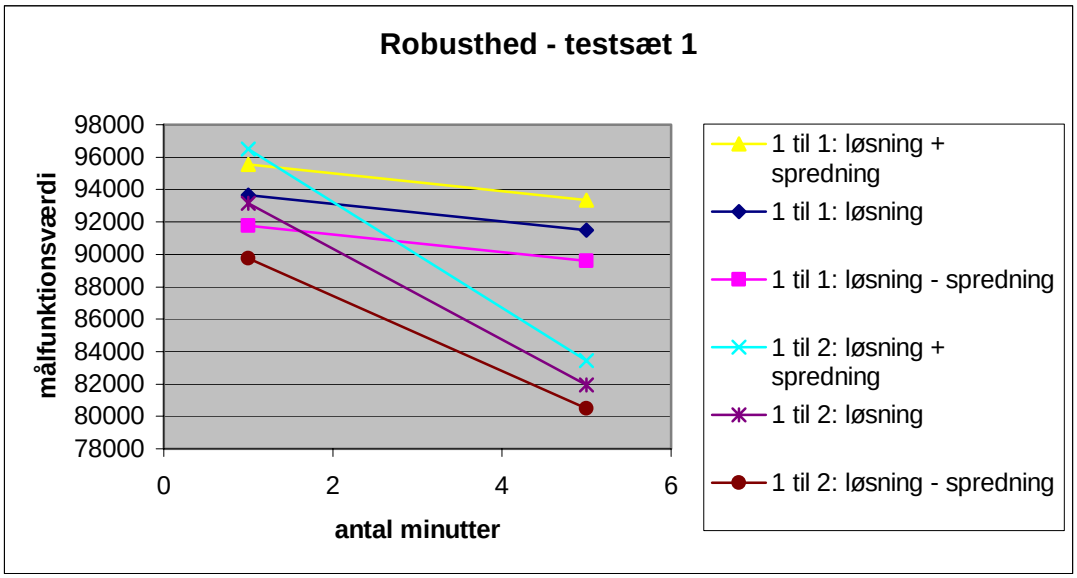
Ud fra ovenstående testsæt vil jeg nu foretage en lang række eksperimenter, for at fastlægge de vigtigste parametre for algoritmen. Jeg vil starte med at teste robustheden af algoritmen, ved at køre samme test 5 gange i træk, med samme startbetingelser og sammenligne løsningerne. Med robusthed menes om algoritmen ved flere kørsler med samme startbetingelser producerer løsninger, hvis målfunktionsværdi ligger indenfor samme interval. En meget robust algoritme vil returnere meget ens resultater ved hver kørsel, med samme startbetingelser, mens en urobust algoritme vil returnere resultater, der er meget forskellige fra hinanden.

7.3.1 Robusthed

I dette afsnit vil jeg vise, hvilke udsving algoritmen har ved 5 kørsler af samme initielle løsning. Selv i det ideelle tilfælde, er det ikke meningen, at algoritmen skal give den samme løsning ved flere kørsler, på nær hvis det er optimum, da dele af algoritmen er baseret på tilfældighed. Hvis jeg kunne få optimum 5 gange i træk, ville det selvfølgeligt være så godt som muligt. Realistisk vil jeg gerne minimere udsvinget i løsningerne baseret på samme startbetingelser. Her antages at tilfældigheden, i form af random() kald i programmet, ikke giver identiske værdier. Denne antagelse er også opfyldt i C# så vidt jeg er informeret. Grunden til jeg har et afsnit om robusthed, inden den endelige tuning af de vigtigste parametre, er for at se om udsving i de kommende forsøg kan tilskrives ren tilfældighed eller om det er faktiske ændringer. Desuden er det for at se den generelle

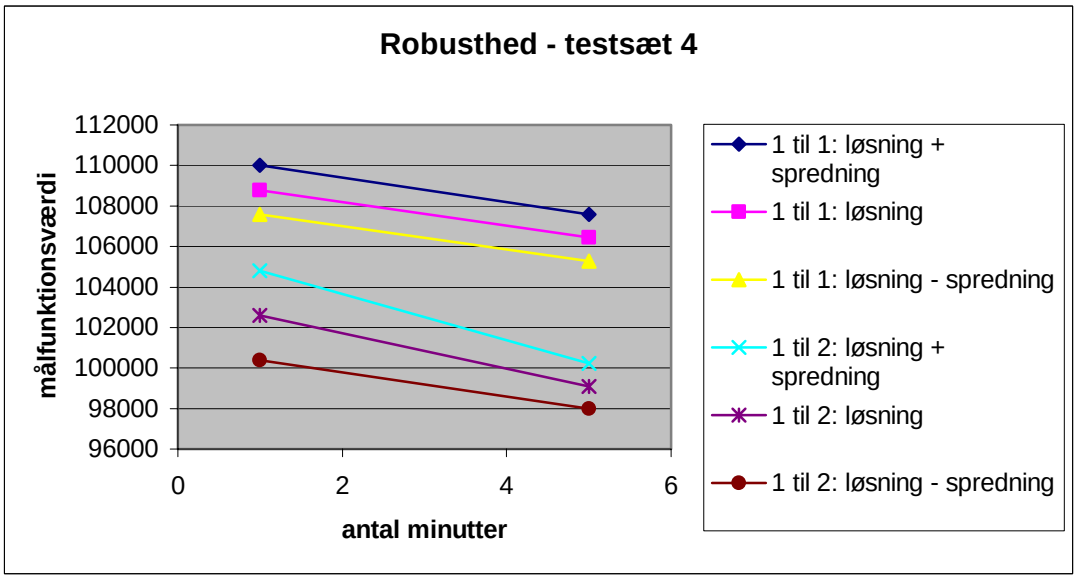
tendens for de to typer af ombytning. I de senere tests vil hvert punkt være aftegnet, så spredningen også fremgår efter den endelige tuning.

Hvert punkt i figurerne herunder er middelværdien af 5 kørsler af henholdsvis 1 til 1 ombytning og 1 til 2 ombytning. Disse tests er før dynamisk indstilling af parametre blev implementeret og terminalindsættelse og konsolideringsfunktionen bliver kaldet på faste tidspunkter indenfor 2000*√(antal minutter) iterationer. Spredningen er udregnet ud fra de 5 kørsler og er vist på figurerne.



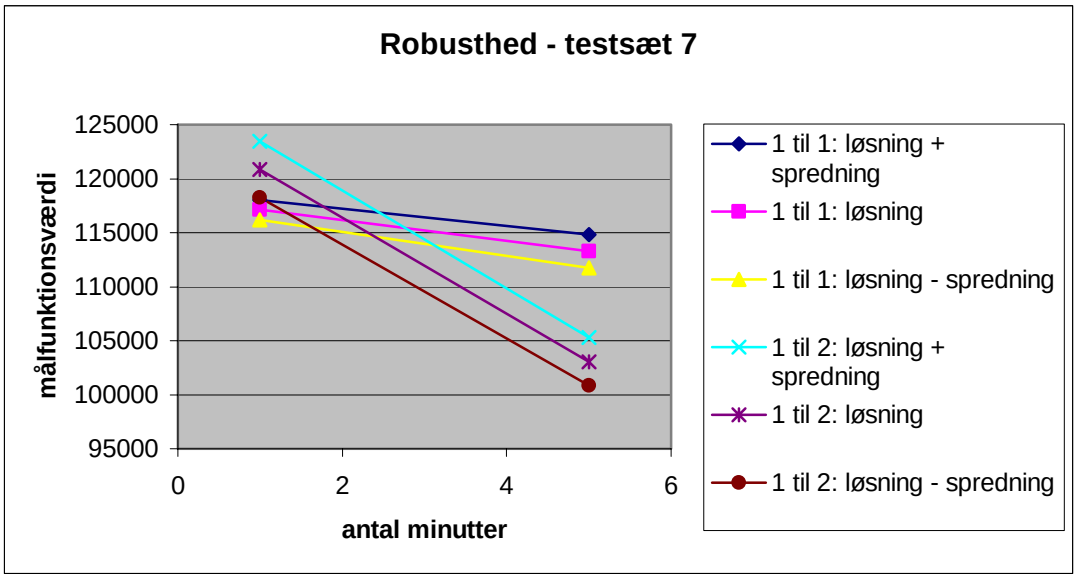
Figur 42 – Robusthed for testsæt 1

For testsæt 1 ligner løsningerne ved 1 minut hinanden en del, dog har 1 til 2 ombytningen en lidt større spredning end 1 til 1 ombytning. Efter 5 minutter skiller 1 til 2 ombytning sig tydeligt ud, ved at have en klart forbedret målfunktionsværdi og en mindre spredning end 1 til 1 ombytning.



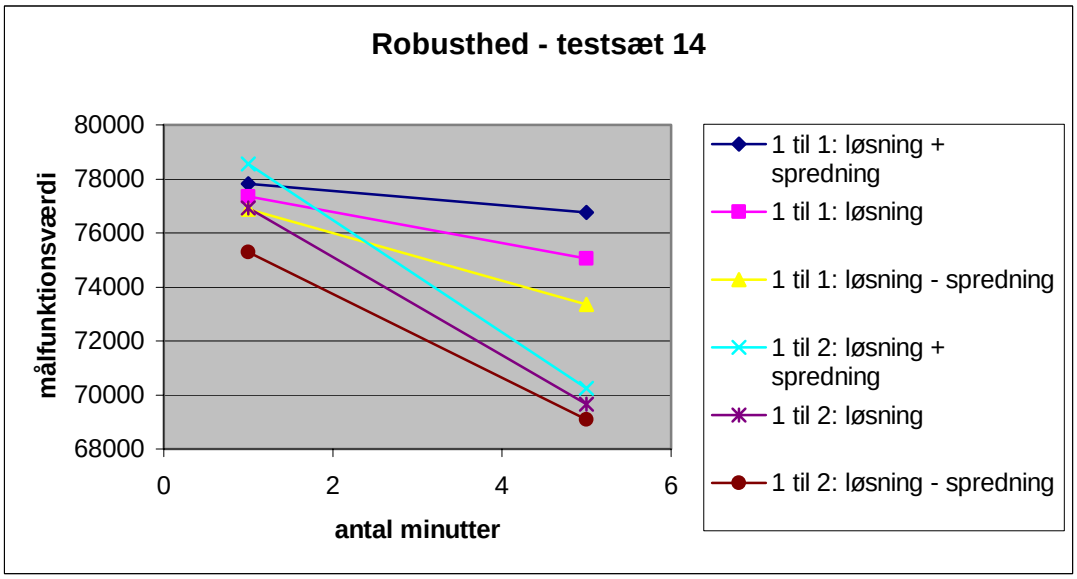
Figur 43 – Robusthed for testsæt 4

For testsæt 4 er 1 til 2 ombytning betydeligt bedre end 1 til 1 ombytning, både ved 1 og 5 minutter. 1 til 1 ombytning udviser igen en meget kontinuert forbedring med ca. samme spredning, mens 1 til 2 ombytning har mere markante forbedringer og en faldende spredning.



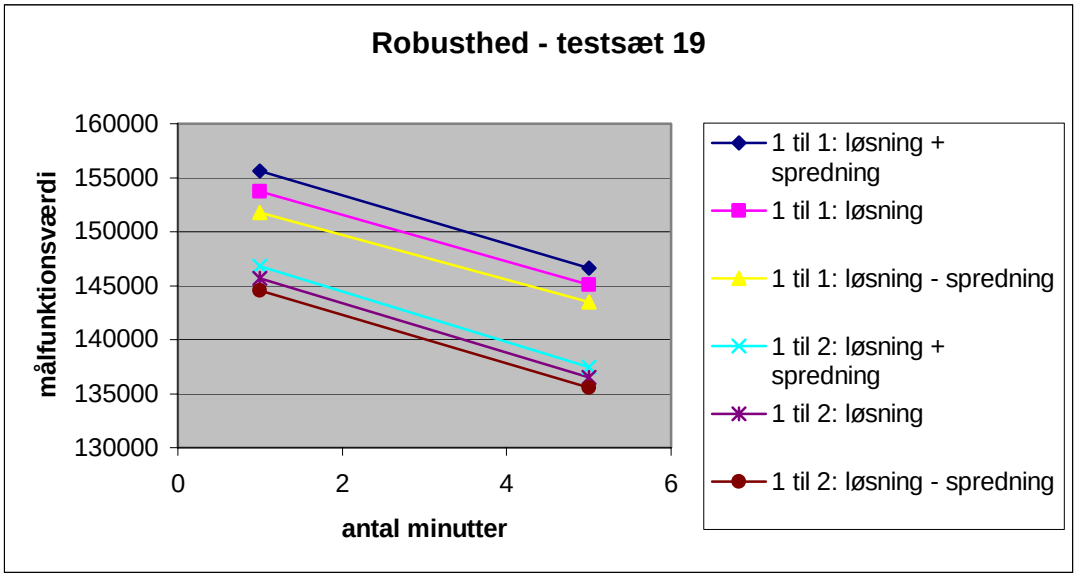
Figur 44 – Robusthed for testsæt 7

For testsæt 7 gælder nogenlunde det samme, som for testsæt 1. Igen starter 1 til 1 ombytning bedst, både hvad angår målfunktionsværdi og spredning. Ved 5 minutter er 1 til 2 ombytning dog klart bedst, hvad angår målfunktionsværdi, men har lidt større spredning end 1 til 1 ombytning.



Figur 45 – Robusthed for testsæt 14

Modsat tidligere eksempler har 1 til 2 ombytning både klart bedre målfunktionsværdi og spredning ved 5 minutter for testsæt 14. Ved 1 minut minder det mere om de tidligere sæt.



Figur 46 – Robusthed for testsæt 19

For testsæt 19 er dommen klar. 1 til 2 ombytning klarer sig bedst på alle punkter, selvom målfunktionsværdien kun bliver forbedret i samme tempo som for 1 til 1 ombytning.

Sammenfattes værdierne for målfunktionværdiernes gennemsnit og spredning, kan en figur som nedenstående konstrueres:

testsæt	Robusthed tid i minutter	1 til 1 ombytning 1	1 til 1 ombytning 5	1 til 2 ombytning 1	1 til 2 ombytning 5
1	gennemsnit	93664.3	91473.33	**93130.26**	**81952.14**
	spredning	**1883.223**	1876.18	3373.761	**1485.351**
4	gennemsnit	108793.9	106431.2	**102581.4**	**99100.81**
	spredning	**1209.319**	1159.295	2209.107	**1126.088**
7	gennemsnit	**117124.9**	113303.9	120845.6	**103055.2**
	spredning	**913.0496**	**1544.49**	2588.7	2217.697
14	gennemsnit	**77359.16**	75050.66	76924.6	**69666.59**
	spredning	**473.8204**	1710.099	1643.225	**572.2408**
19	gennemsnit	153698.3	145068.8	**145689.9**	**136525.9**
	spredning	1935.314	1561.919	**1124.102**	**942.0891**
	antal markerede	**6**	1	4	**9**

Figur 47 – Opsummering af resultaterne for robusthed

Havde jeg haft målinger ved 15 minutter var konklusionen højst sandsynligt mere entydig. Med de nuværende data er det dog stadigt muligt at drage en konklusion. 1 til 1 ombytning har ved 1 minuts kørsel 6 markerede resultater mod 4 for 1 til 2 ombytning. Ved 5 minutter er 1 til 2 ombytning klart bedst, kun 1 gang har 1 til 1 ombytning en bedre spredning. Antager man at tendensen fortsætter og 1 til 2 ombytning forbedrer sig mere jo længere tid der går, i forhold til 1 til 1 ombytning, så er det klart at 1 til 2 ombytning har den bedste robusthed af de to typer ombytning.

Robusthed tid i minutter	1 til 1 ombytning 1	1 til 1 ombytning 5	1 til 2 ombytning 1	1 til 2 ombytning 5
gennemsnitlig spredning	1282.945	1570.397	2187.779	1268.693

Figur 48 – Opsummering af spredning for 1 til 1 og 1 til 2 ombytning

7.3.2 Dynamisk indstilling af parametre

Det er blevet klart, at princippet fra [Cor01] med god grund kan anvendes på andre beslutningsprocesser i mit projekt. Jeg benytter flere forskellige nabolag (bla. hjælpefunktionerne) og valget mellem, hvornår man skal skifte til et nyt nabolag, er oplagt at lave til en dynamisk proces. Her bliver succesfulde skift belønnet ved at forekomme oftere. Modsat bliver dårlige skift straffet ved at forekomme sjældnere. Om et skift er godt eller dårligt bestemmes ud fra målfunktionsværdien.

Den grundlæggende ide i dynamisk indstilling af parametre er, at en startværdi automatisk bliver justeret til en bedre værdi, hvis det er nødvendigt. Det er specielt valget af, ved hvilken iteration indsættelse af terminal (NT) eller konsolideringsfunktionen (NK) skal ske. Ovenfor i undersøgelsen af robusthed blev NT kaldt, hver gang antal iterationer modulo 2000*√(antal minutters total kørselstid) var lig 999. NK blev kaldt, hver gang antal iterationer modulo 2000*√(antal minutters total kørselstid) var lig 1999. Jeg havde tidligere forsøgt med andre indstillinger, men ingen virkede særligt godt.

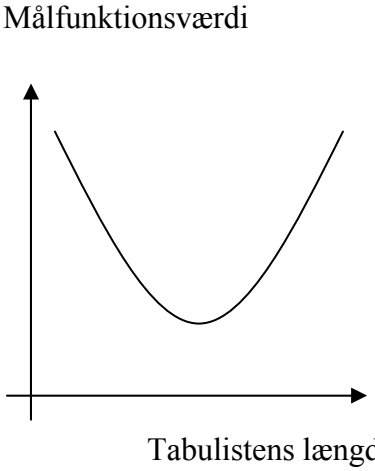
Ved dynamisk indstilling af parametre bliver en given funktion kaldt, når et vist antal iterationer er forløbet uden en global forbedring. Kalder vi dette antal count, vil funktionen blive kaldt, første gang antal iterationer uden forbedring bliver større end count. Herefter henholdsvis sænkes eller hæves antal iterationer for kald af funktionen, alt efter om kaldet af funktionen gav en bedre eller værre målfunktionsværdi end den foregående. Dette betyder at parameteren (count) automatisk bliver indstillet til en ny værdi, der tilsvarer succesen fra sidste kald af funktionen. I det konkrete tilfælde henholdsvis fordobles count ved en forbedret målfunktionsværdi og halveres ved en dårligere målfunktionsværdi end den foregående. I tilfælde af en forværring af målfunktionsværdien vil antal iterationer nu være dobbelt så stor som count, da denne lige er blevet halveret. I næste iteration vil algoritmen derfor kalde funktionen igen. Det er ikke hensigtsmæssigt at køre f.eks. konsolideringsfunktionen flere gange i træk, da den anden kørsel ikke kan bidrage med nye ting, hvorfor denne opførsel må undgås. Derfor er der en sikkerhedsforanstaltning, der garanterer, at en funktion ikke kan kaldes flere gange i træk. Hvis en global forbedring sker, bliver antal iterationer uden forbedring nulstillet og dermed vil en funktion først blive kaldt igen, når antallet kommer over count.

I det følgende vil count for indsættelse af terminal blive vist som NT=<tal>, count for konsolideringsfunktionen som NK=<tal> og sikkerhedsforanstaltningen som limit=<tal>. De viste tal er startværdien såfremt andet ikke er anført.

7.3.3 Tabulisten

Ideen med at tilføje en ombytning, typisk også kaldet en overgang, til tabulisten er direkte inspireret fra litteraturen om tabu søgning. Det bliver bla. beskrevet af [Lau94] og benyttet af [You01]. Løsningsrummet for problemet kan betragtes som en bølget flade i rummet. På fladen er der bakker og dale og højden repræsenterer målfunktionsværdien. Målet for en tabu søgning er at finde det laveste punkt på fladen, dvs. den mindste målfunktionsværdi, såfremt vi gerne vil minimere målfunktionsværdien. En løsning svarer til et punkt på fladen og nabolaget er området omkring punktet. Tabu søgning prøver altid at søge nedad hvis det er muligt. Ender søgningen i et lokalt minimum, skal tabulisten hjælpe til med at udelukke de løsninger, der prøver at holde søgningen nede i det lokale minimum.

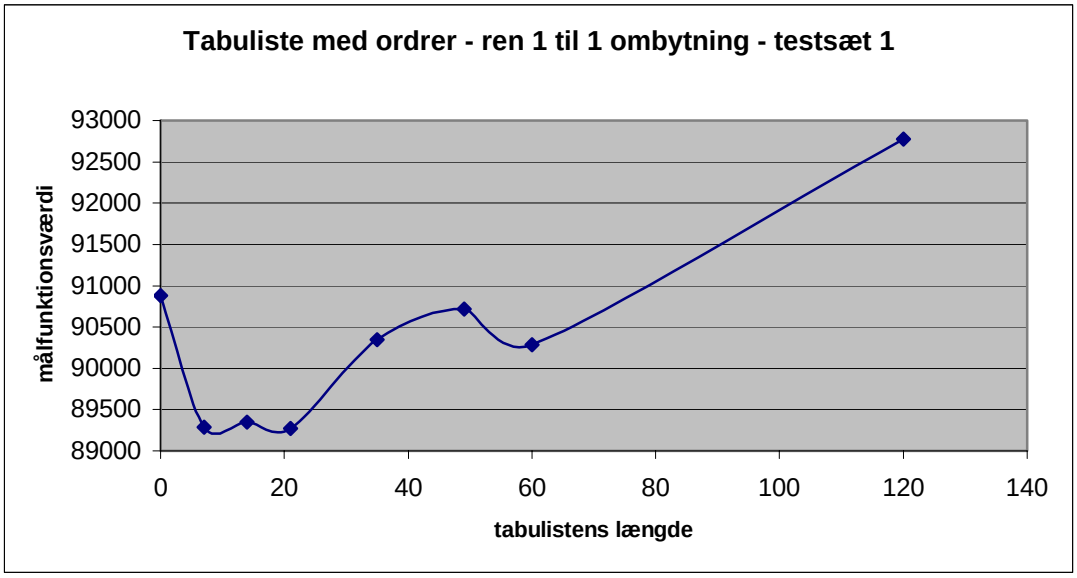
Hvis tabulisten er for kort er der større sandsynlighed for, at algoritmen kommer til at udvise cyklisk opførsel og ikke finder ned til det globale minimum. Hvis tabulisten er for lang, bliver algoritmen hæmmet i dens bevægelighed. Har tabulisten den rette længde, er det sandsynligt at cyklisk opførsel kan undgås, samtidigt med at algoritmen ikke er unødvendigt hæmmet af tabulisten.
Så man kunne antage, at målfunktionsværdien afhang af tabulistens længde som et positivt polynomium, som vist på Figur 49.



Figur 49 – Antaget sammenhæng mellem målfunktionsværdien og tabulistens længde

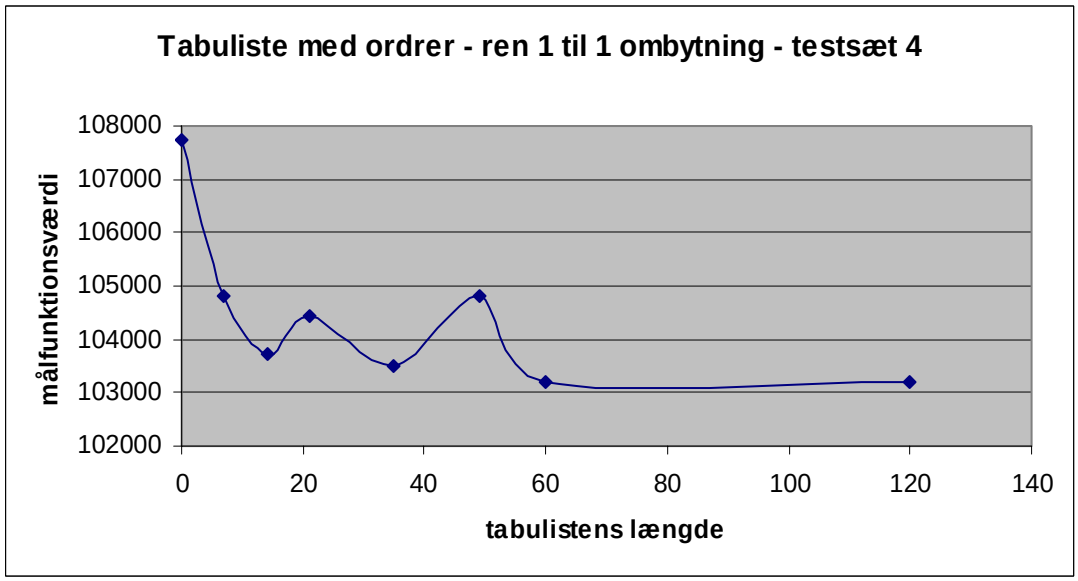
Denne sammenhæng mellem målfunktionsværdi og tabulistens længde har jeg testet eksperimentelt, for at finde den bedste værdi for tabulistens længde. Herunder er de 5 testsæt med 1 til 1 ombytning med varierende længde af tabulisten.

Konsolideringsfunktionen er i de 5 tests slået fra for at give et indblik i en ren 1 til 1 ombytning. For alle 5 grafer gælder at tabulisten indeholder sæt af ordrer, som tidligere beskrevet. Den blå linie, der forbinder hvert punkt, er kun beregnet til at give lettere overblik over punkterne og deres sammenhæng. Hvert punkt repræsenterer et gennemsnit af 3 kørsler af 5 minutter per styk.



Figur 50 – Tabuliste med ordrer – ren 1 til 1 ombytning – testsæt 1

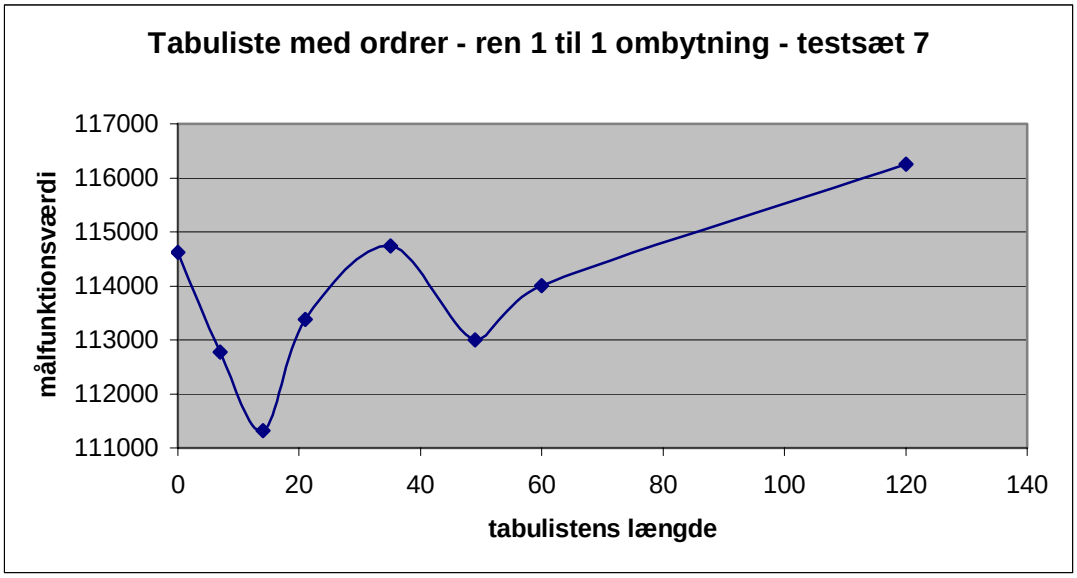
For testsæt 1 på Figur 50 haves den forventede sammenhæng mellem målfunktionsværdi og tabulistens længde, selvfølgeligt med en del variationer. Der vil altid være en vis portion tilfældighed i resultatet, da algoritmen indeholder elementer af tilfældighed. Punkterne på grafen svarer til en længde af tabulisten på 0, 7, 14, 21, 28, 35, 49, 60 og 120. Det bemærkes, at en længde af tabulisten på mellem 7 og 21 giver de bedste resultater. Ved en længde af tabulisten på 1200 fik jeg værdien 95325, der bekræfter stigningen man ser hen mod de 120. Kørslen, for en længde af tabulisten på 1200, er foretaget efter, grafen blev lavet, hvorfor resultatet ikke er med på grafen. Desuden repræsenteres resultaterne mellem 0 og 60 dårligt, hvis x aksen går fra 0 til 1200+.



Figur 51 – Tabuliste med ordrer – ren 1 til 1 ombytning – testsæt 4

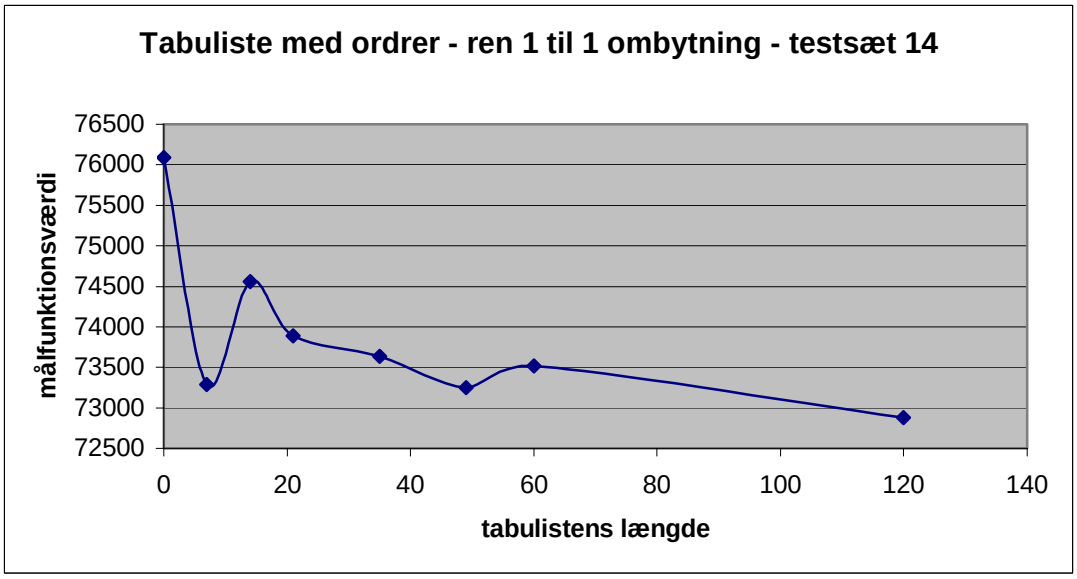
Testsæt 4 på Figur 51 udviser en noget anderledes opførsel og lader til at få bedre målfunktionsværdi ved større længder af tabulisten. Ved 1200 fik jeg værdien 107008, hvilket er på

niveau med værdien ved 0, så målfunktionsværdien bliver ikke ved med at falde, jo længere tabulisten bliver.



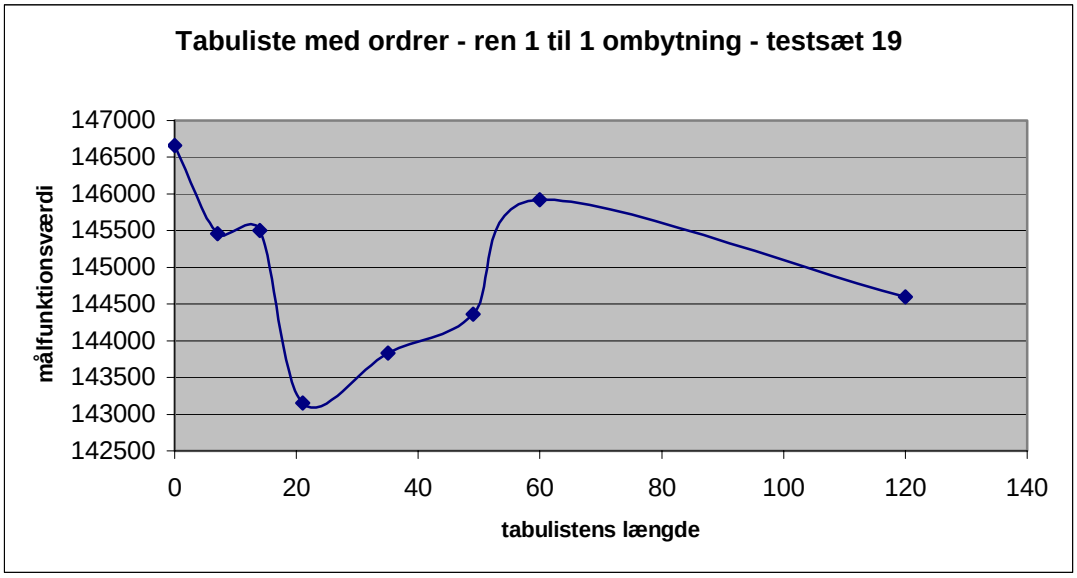
Figur 52 – Tabuliste med ordrer – ren 1 til 1 ombytning – testsæt 7

For testsæt 7 på Figur 52 er den bedste målfunktionsværdi ved en længde på 14 og herefter stiger målfunktionsværdien. Ved 1200 er den oppe på 120143.



Figur 53 – Tabuliste med ordrer – ren 1 til 1 ombytning – testsæt 14

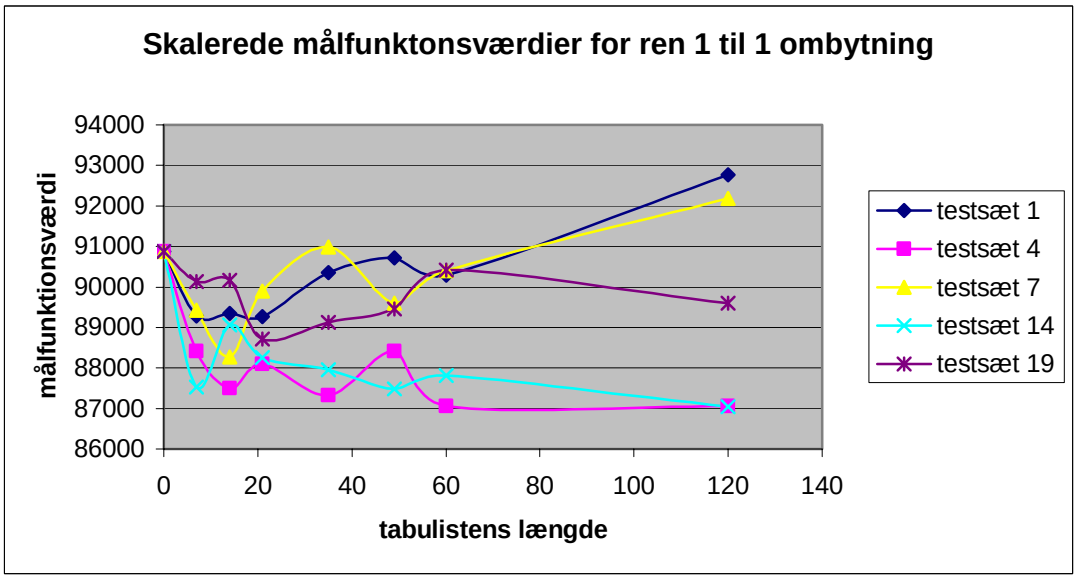
Testsæt 14 lader til at give bedre resultater, jo længere tabulisten er og ved 1200 er målfunktionsværdien 73521.



Figur 54 – Tabuliste med ordrer – ren 1 til 1 ombytning – testsæt 19

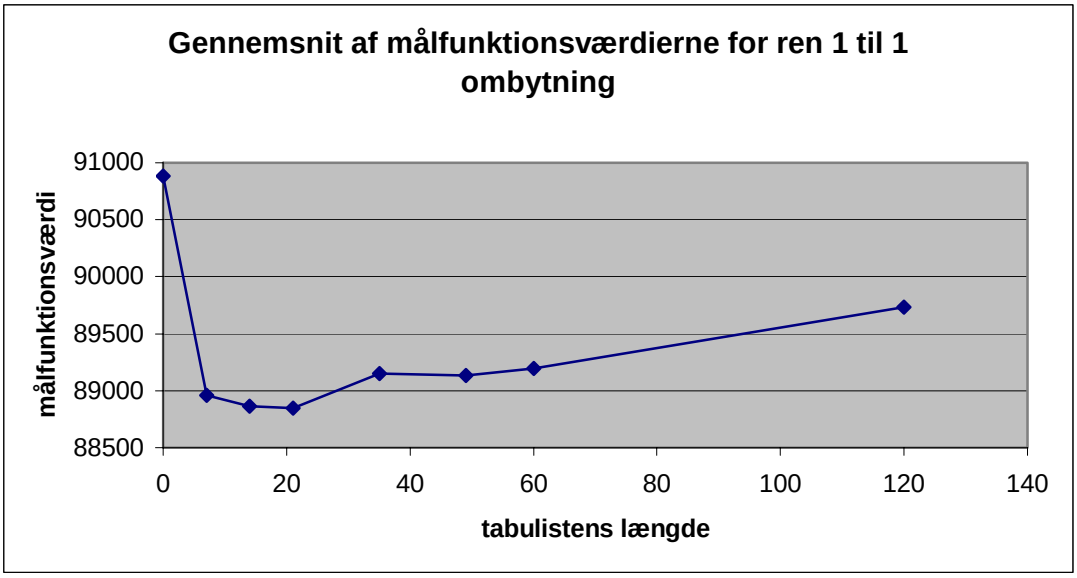
Testsæt 19 har indenfor de første målepunkter op til 60, nogenlunde den forventede opførsel, med den bedste værdi ved 21. Men ved 120 er målfunktionsværdien faldet i forhold til ved 60 og ved 1200 er den 145995, altså en lille stigning til samme niveau som 60.

Sammenfatter jeg ovenstående 5 figurer, ved at skalere målfunktionsværdierne så de starter i samme punkt for en længde på 0, bliver resultatet som vist på Figur 55. Umiddelbart er det ikke let, at komme med en entydig konklusion.



Figur 55 – Skalerede målfunktionsværdier for ren 1 til 1 ombytning

Konstrueres i stedet en graf, der viser gennemsnittet af målfunktionsværdierne ved hvert målepunkt, er konklusionen mere entydig.

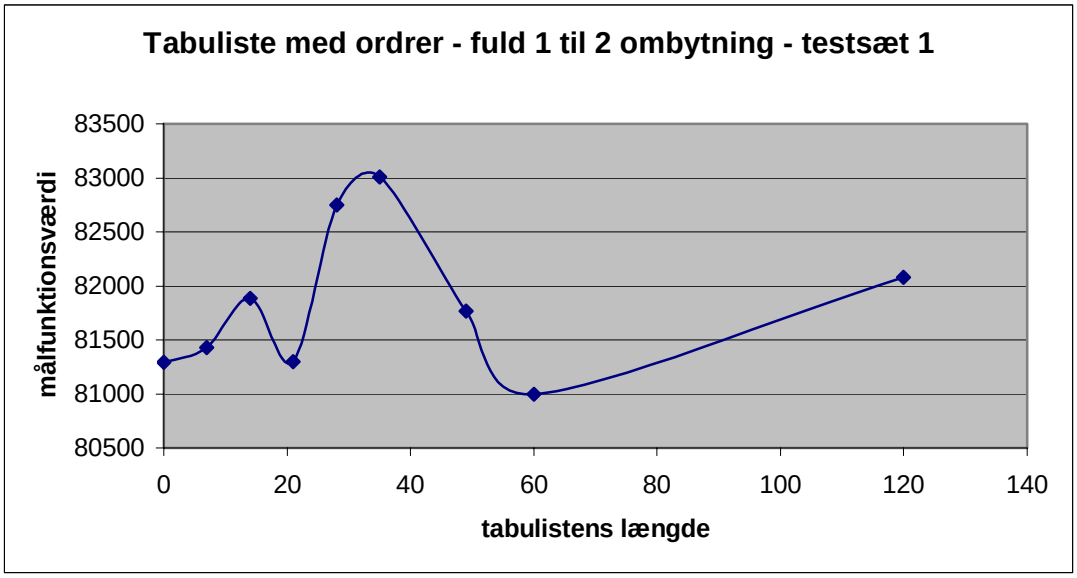


Figur 56 – Gennemsnit af målfunktionsværdier for ren 1 til 1 ombytning, skaleret.

Selvom der ikke kan formuleres en fuldstændig entydig konklusion af ovenstående figurer, er den generelle tendens, at en længde af tabulisten mellem 7 og 21 give gode resultater. Især Figur 56 viser dette og udpeger en længde af tabulisten på 21 som den bedste værdi, med en meget lille margin i forhold til 7 og 14. Desuden opfører grafen på Figur 56 sig i grove træk, som jeg havde antaget. Tilføjes konsolideringsfunktionen til 1 til 1 ombytning, må det påvirke resultatet på en eller anden måde. Hvordan resultatet påvirkes har jeg ikke haft tid til at teste, men det kunne helt sikkert være interessant.

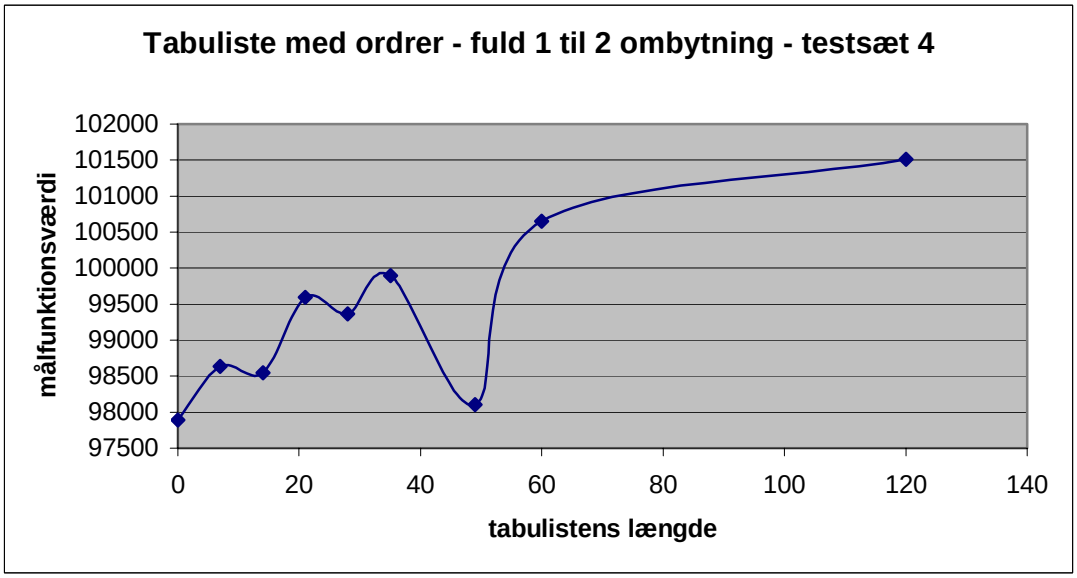
Går vi over til 1 til 2 ombytning, er der mange elementer, der spiller ind, i forhold til den noget simplere 1 til 1 ombytning. Der er både geografisk afgrænsning og indsættelse af terminal, udover konsolideringsfunktionen der er fælles. Især indsættelse af terminal må forventes at ændre resultaterne betydeligt, da løsningsrummet bliver ændret voldsomt i forhold til en normal ombytning. For at understrege dette har jeg kaldet det for fuld 1 til 2 ombytning, modsat ren 1 til 1 ombytningen ovenover, hvor konsolidering er deaktiveret.

For nedenstående tests gælder at NK=144, NT=121 og limit=337.



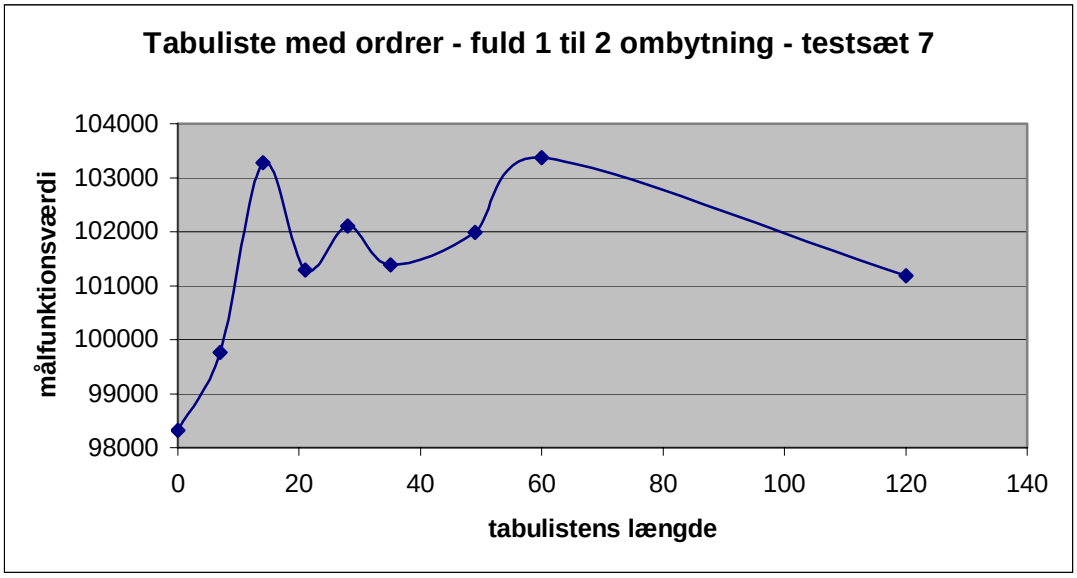
Figur 57 – Tabuliste med ordrer – fuld 1 til 2 ombytning – testsæt 1

Testsæt 1 giver kun anledning til at konkludere, at målfunktionsværdien for 1 til 2 ombytning, ikke afhænger af tabulistens længde på en simpel måde. Umiddelbart er den bedste værdi ved en længde på 60.



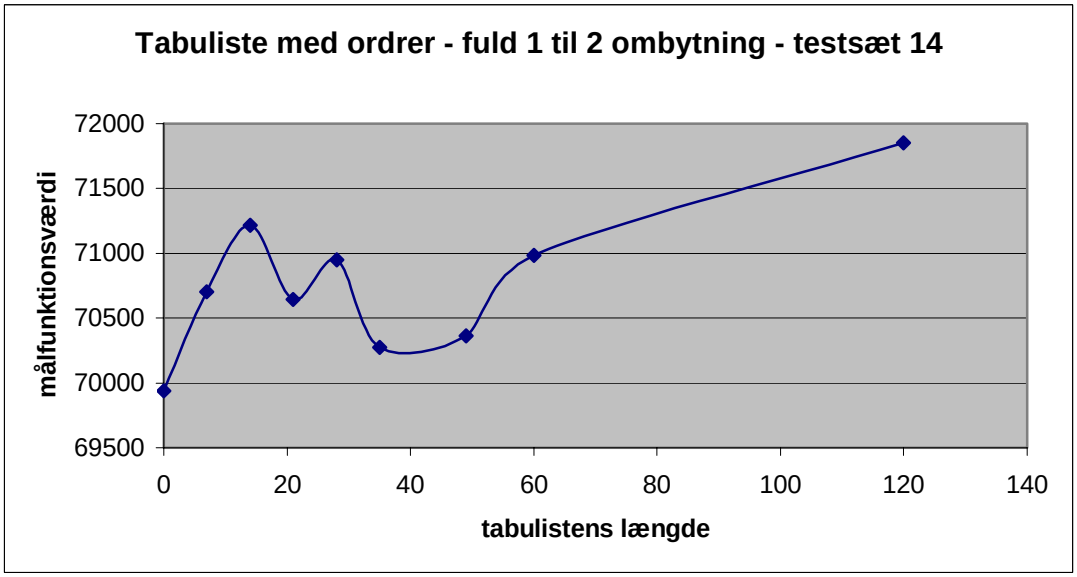
Figur 58 – Tabuliste med ordrer – fuld 1 til 2 ombytning – testsæt 4

En lidt anderledes tendens ses for testsæt 4, hvor en længde på 0 giver det bedste resultat og udover punktet ved 49 er der en stigende tendens.



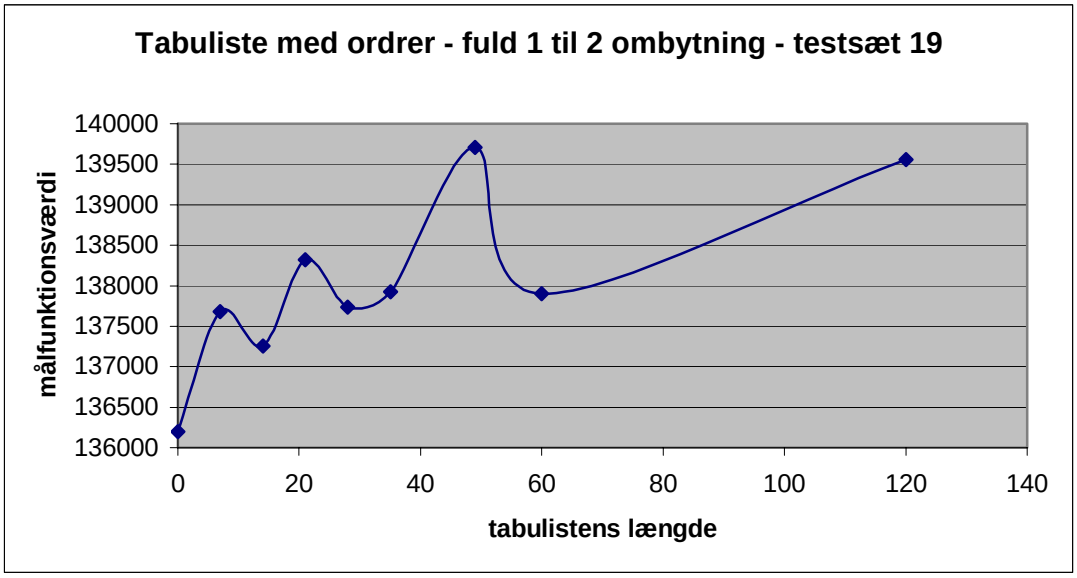
Figur 59 – Tabuliste med ordrer – fuld 1 til 2 ombytning – testsæt 7

Testsæt 7 har klart den bedste målfunktionsværdi ved en længde på 0. Noget kunne tyde på, at tabulistens funktion er mere hæmmende for 1 til 2 ombytning, med alt hvad det indebærer i forhold til 1 til 1 ombytning.



Figur 60 – Tabuliste med ordrer – fuld 1 til 2 ombytning – testsæt 14

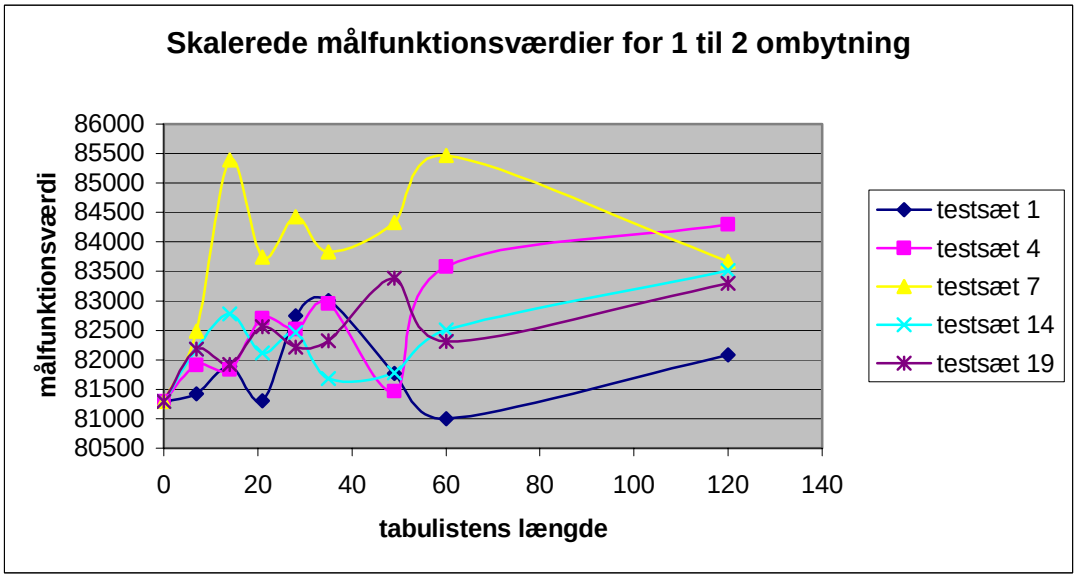
Igen giver en længde på 0 den bedste målfunktionsværdi, mens resten af målepunkterne ligger på et højere niveau.



Figur 61 – Tabuliste med ordrer – fuld 1 til 2 ombytning – testsæt 19

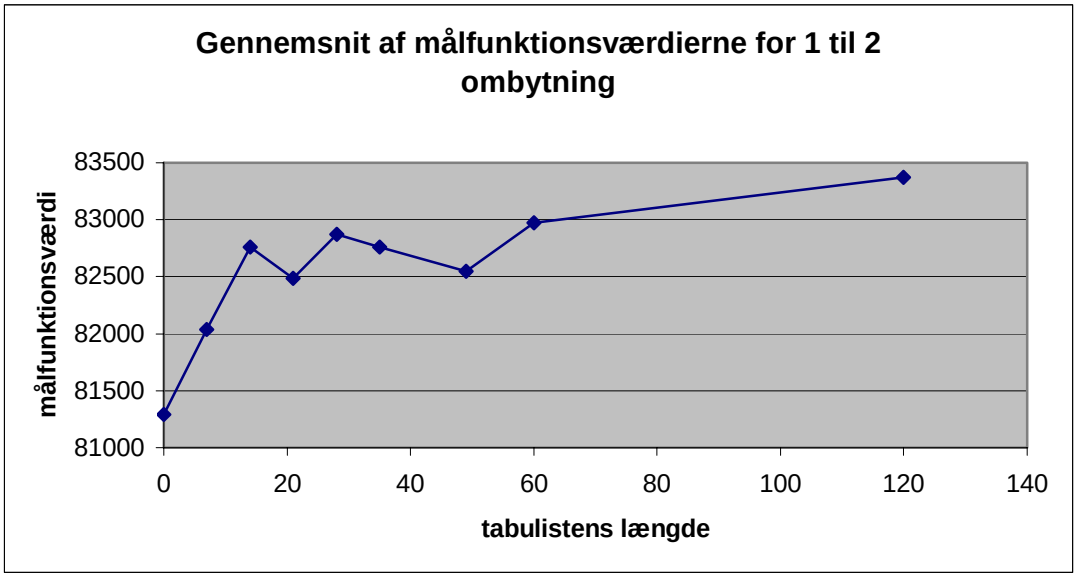
Testsæt 19 gør det klart, at en længde på 0 giver den bedste målfunktionsværdi og større længder har en tendens til at give dårligere målfunktionsværdier.

Sammenfatter vi de 5 testsæt ved at skalere dem, så udgangspunktet er det samme, fås en figur som den på Figur 62. Selvom en længde på 60 kan præsentere et enkelt resultat bedre end en længde på 0, er det allerede nu klart, at en længde på 0 giver de bedste løsninger for 1 til 2 ombytning.



Figur 62 – Skalerede målfunktionsværdier for fuld 1 til 2 ombytning

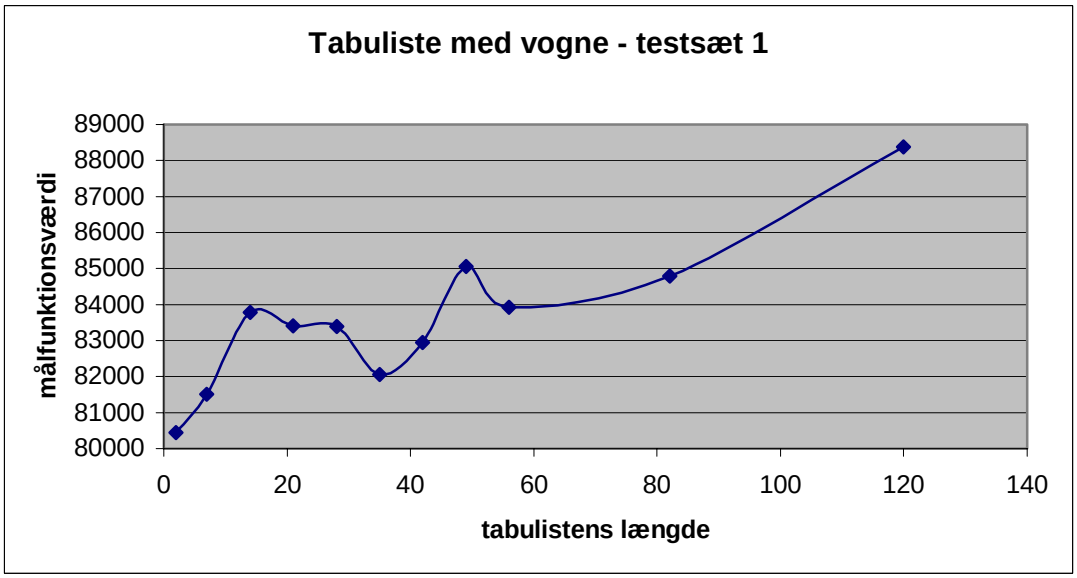
Summerer vi over målfunktionsværdierne, giver det en graf som vist nedenunder på Figur 63. Det er tydeligt, at en tabuliste med længden 0 giver de bedste resultater for 1 til 2 ombytning.



Figur 63 – Gennemsnit af målfunktionsværdier for fuld 1 til 2 ombytning, skaleret

Grunden til, at tabulisten ikke har en gavnlig funktion for 1 til 2 ombytning, er ikke umiddelbar klar, men jeg har en formodning om, at især indsættelse af terminal deformerer løsningsrummet på en uhensigtsmæssig måde i forhold til en tabu søgning. Det kunne kontrolleres ved at køre 1 til 2 ombytning uden indsættelse af terminal, men det er der desværre ikke tid til.

Som en ekstra test prøvede jeg at benytte vognene og dermed ruterne, som elementer på tabulisten. Hver gang en ombytning er sket mellem to vogne, bliver begge vogne tilføjet til tabulisten, som et sæt. Nedenstående graf er lavet vha. 1 til 2 ombytning på testsæt 1. Hvert punkt repræsenterer en kørsel af 5 minutters varighed, dvs. det er ikke et gennemsnit af flere kørsler.



Figur 64 – Tabuliste baseret på ruter / vogne

Ovenstående graf giver samme konklusion som Figur 63. Jeg er blevet gjort opmærksom på, at en længere tabuliste, som tidligere beskrevet, kunne hæmme algoritmen. Dermed kunne et faldende

antal iterationer være skyld i den dårligere målfunktionsværdi ved længere tabulister. For ovenstående graf bliver tabulisten dog ikke lang nok til at påvirke antallet af iterationer betydeligt. Således ligger antallet af iterationer tilfældigt spredt mellem ca. 3400 og 3700 for ovenstående graf.

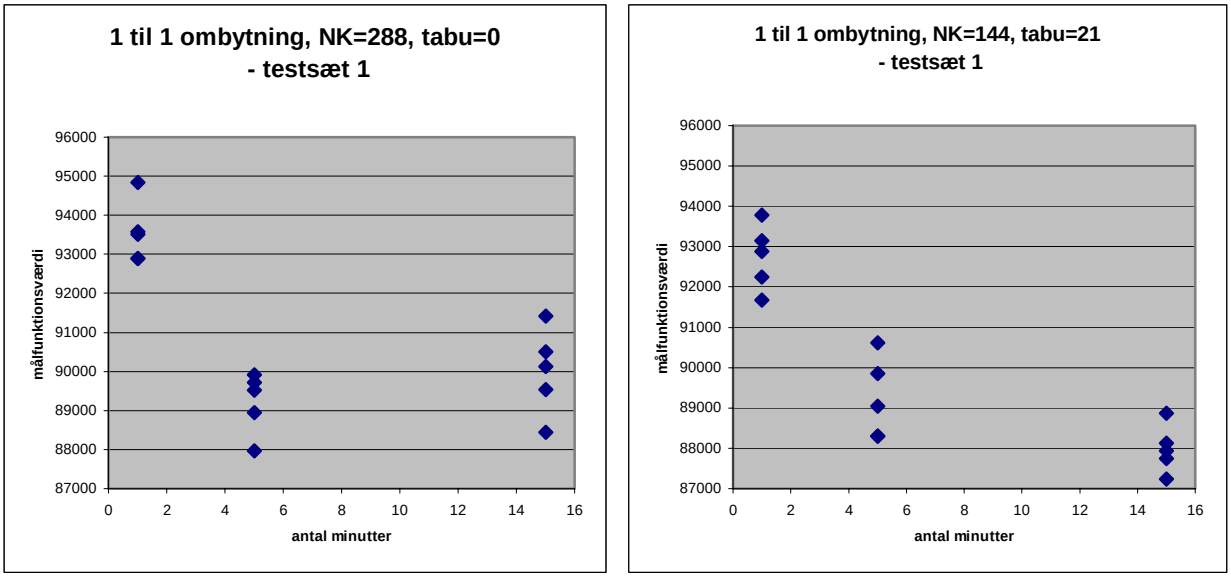
Ovenstående eksperimenter kan sammenfattes i følgende konklusioner:

- Ren 1 til 1 ombytning klarer sig bedst ved en tabuliste med længde 21.
- 1 til 2 ombytning klarer sig bedst ved en tabuliste med længde 0.
- Jeg vil fortsætte med tabulister baseret på ordrer, da det største testmateriale ligger her, dvs. jeg vil ikke fortsætte med tabulister baseret på vogne eller ruter.
- De værdier for NK og NT jeg har afprøvet, ligger på et niveau hvor de både justeres op og ned. Jeg antager derfor at de er udmærket, da der ikke er tid til en tilbundsgående undersøgelse. Jeg har ingen direkte sammenligninger med konstante værdier for NK eller NT.

7.3.4 Valg af ombytning

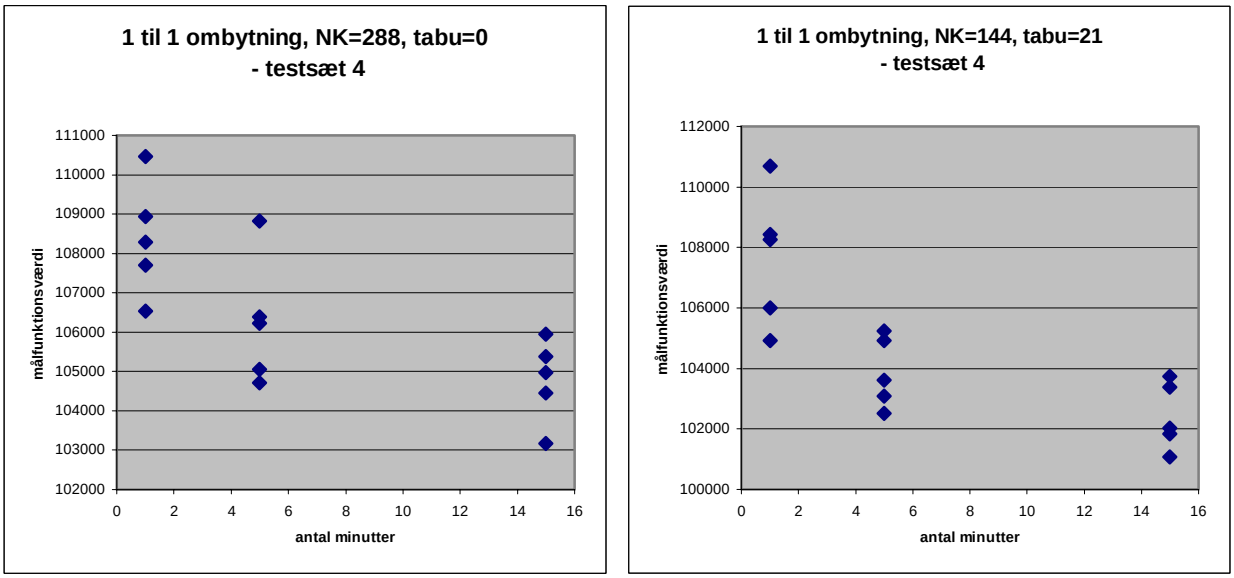
Foregående afsnit giver anledning til 2 forskellige eksperimenter, der skal vise, om valget af ovenstående værdier rent faktisk giver de forventede resultater. Disse 2 eksperimenter vil afprøve henholdsvis 1 til 1 ombytning og 1 til 2 ombytning med de foreslåede parametre. Da 1 til 1 ombytning mangler værdier for NK, NT og limit vil jeg foreslå nogle.

Det første eksperiment vil benytte en tabuliste med længde 0 og NK=288, NT=216 og limit=537. Det andet eksperiment vil benytte en tabuliste med længde 21 og NK=144, NT=121 og limit=337. Begge eksperimenterne vil blive kørt med ombytningerne 1 til 1 og 1 til 2, med alt hvad det inkluderer. Nedenstående grafer er opdelt i to søjler, hvor første søjle er det første eksperiment, mens anden søjle er det andet eksperiment. Denne opstilling er for at lette sammenligningen af de eksperimenter.
Begge eksperimenter vil blive udført ved 15 kørsler hvoraf henholdsvis 5 kørsler er af 1, 5 og 15 minutters varighed. Disse tidspunkter er valgt på baggrund af, hvad jeg forventer er aktuelt for en algoritme af denne type og for at give et billede af, hvilket tempo algoritmen forbedrer målfunktionsværdien. En kørsel på 1 minut må betragtes som en meget kort planlægning. 5 minutters kørselstid er en middel planlægningstid, men skulle give betydelige forbedringer i forhold til 1 minut. 15 minutter er en lang planlægningstid, hvor mange løsninger kan undersøges. Modsat disse tests kunne man også forestille sig, at algoritmen kørte i meget lang tid, f.eks. igennem en hel nat, men det vil jeg ikke benytte i denne testfase.



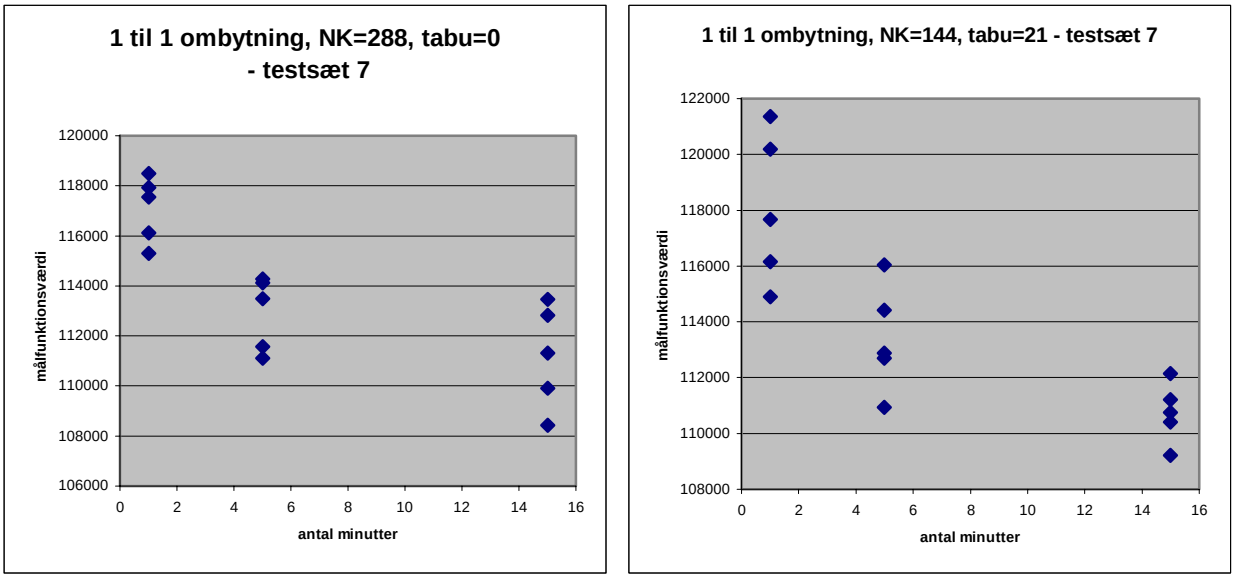
Figur 65 a og b – 1 til 1 ombytning – testsæt 1

På Figur 65 sammenlignes testsæt 1 for en 1 til 1 ombytning med NK=288 og en tabuliste med længde 0 mod en 1 til 1 ombytning med NK=144 og en tabuliste med længde 21. I dette tilfælde har de to grafer samme inddeling, men det kan ikke forventes generelt. Figur 65a udviser en højst usandsynlig opførsel, da løsningerne ved 5 minutter er bedre end ved 15 minutter. Hvert punkt markerer den bedste løsning opnået under kørslen, hvorfor en kørsel på 15 minutter burde have 3 gange større sandsynlighed for at få den samme eller en bedre værdi end en kørsel på 5 minutter. Som vi tidligere har konkluderet klarer 1 til 1 ombytning sig bedst ved en tabuliste med længde 21 og det er også tilfældet her. Figur 65b er på flere punkter en forbedring i forhold til Figur 65a og især værdierne for 15 minutters kørsel er flotte.



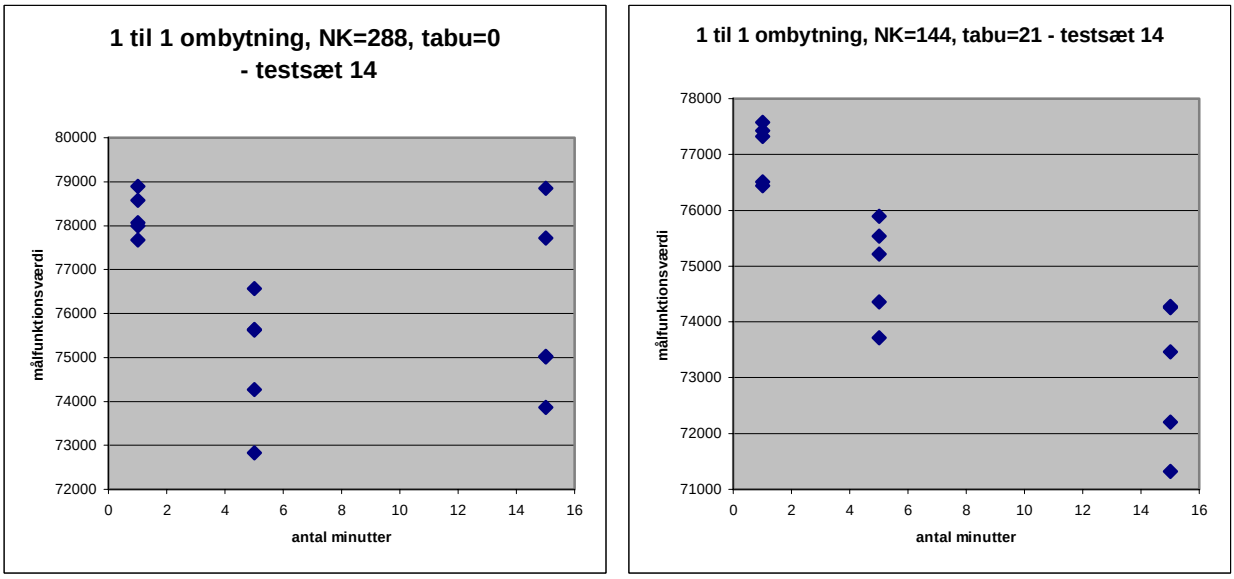
Figur 66 a og b – 1 til 1 ombytning – testsæt 4

For Figur 66a ses en generel forbedring med voksende antal minutter. Tendensen er dog stærkere på Figur 66b, hvor den bedste løsning ved 15 minutter er omkring 2000 bedre end på Figur 66a.



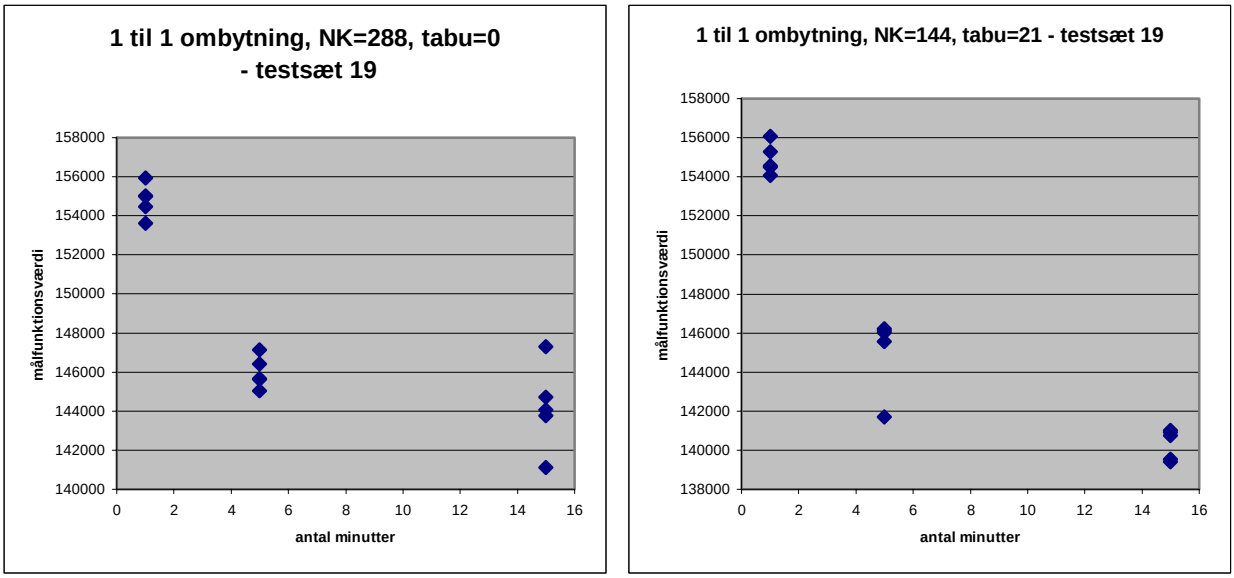
Figur 67 a og b – 1 til 1 ombytning – testsæt 7

På Figur 67 ser vi for første gang en situation, hvor a generelt er bedre end b. Figur 52 der viser sammenhæng mellem målfunktionsværdi og tabulistens længde for testsæt 7 er også den figur, hvor forskellen mellem målfunktionsværdien for 0 og 21 er mindst. Det kan måske forklare, hvorfor figur a klarer sig godt sammenlignet med figur b.



Figur 68 a og b – 1 til 1 ombytning – testsæt 14

På Figur 68 ses en klar forbedring på b i forhold til a. Figur 68a har en endnu mere mærkelig opførsel end Figur 65a. Det er lige før, at det ene punkt ved 15 minutters kørsel er dårligere end det dårligste punkt ved 1 minuts kørsel. Det burde være højst usandsynligt, at den bedste værdi efter 15 minutter er lige så ringe, som en ved 1 minuts kørsel, især når der er mulighed for store forbedringer. Figur 68b har en meget flottere profil og udover at spredningen bliver lidt større, udviser figuren en fin forbedring.



Figur 69 a og b – 1 til 1 ombytning – testsæt 19

På Figur 69a og b er kørslerne ved 1 minut stort set ens. Ved 5 minutter viser figur b generelt bedre resultater og ved 15 minutter er der en klar forbedring og en meget lille spredning.

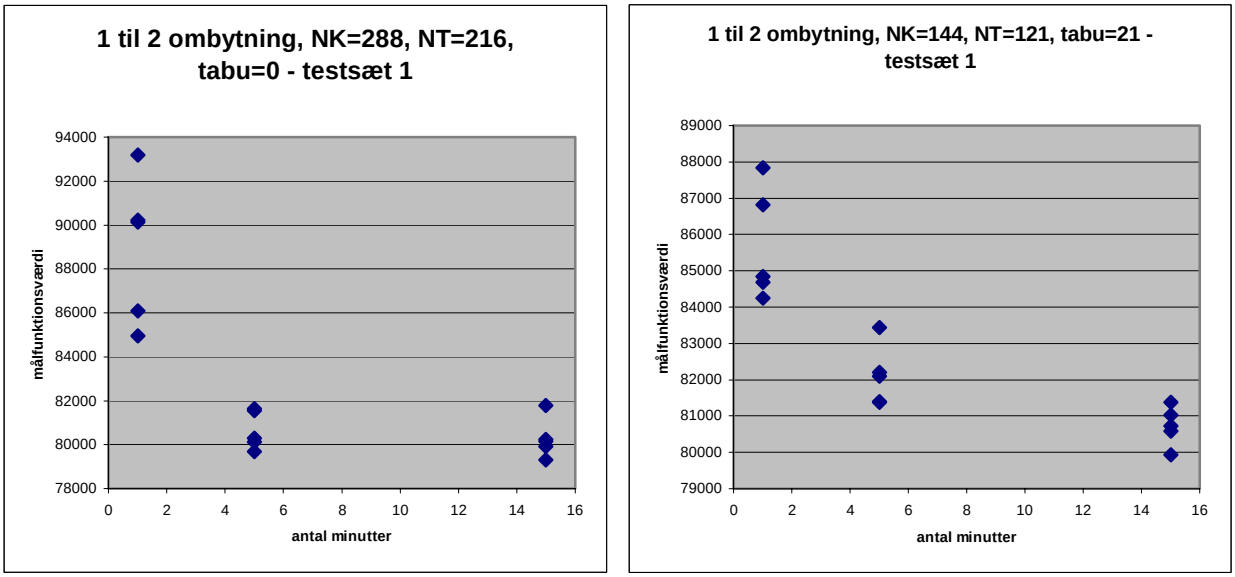
Betragtes alle 10 grafer over 1 til 1 ombytning som en helhed, kan en tabel konstrueres om vist på Figur 70. Her er vist målfunktionsværdierne for de to ombytninger, der sammenlignes, og de bedste målfunktionsværdier i hver kategori er markeret med fed skrift. Målfunktionsværdien ved 'bedste løsning' er den bedste løsning opnået i de 5 kørsler ved et givent tidspunkt. 'Gennemsnit' er den aritmetiske middelværdi for de 5 kørsler. Desuden er antallet af målfunktionsværdier, der er markedet med fed, opsummeret nederst.

datasæt	ombytning tid i minutter	1 til 1, NK=288, tabuliste=0 1	5	15	1 til 1, NK=144, tabuliste=21 1	5	15
1	bedste løsning	92893	**87971**	88443	**91667**	88291	**87243**
	gennemsnit	93543	**89213**	90004	**92745**	89217	**87986**
4	bedste løsning	106529	104712	103167	**104915**	**102516**	**101073**
	gennemsnit	108383	106239	104783	**107658**	**103875**	**102412**
7	bedste løsning	115308	111100	**108419**	**114899**	**110926**	109224
	gennemsnit	**117082**	**112912**	111183	118057	113396	**110744**
14	bedste løsning	77676	**72825**	73861	**76436**	73714	**71324**
	gennemsnit	78241	74983	76091	**77055**	**74943**	**73103**
19	bedste løsning	**153611**	145053	141127	154069	**141705**	**139411**
	gennemsnit	**154804**	145985	144192	154894	**145122**	**140325**
	antal markerede	3	4	1	**7**	**6**	**9**

Figur 70 – Sammenfatning af målfunktionsværdierne for 1 til 1 ombytning med forskellige parametre

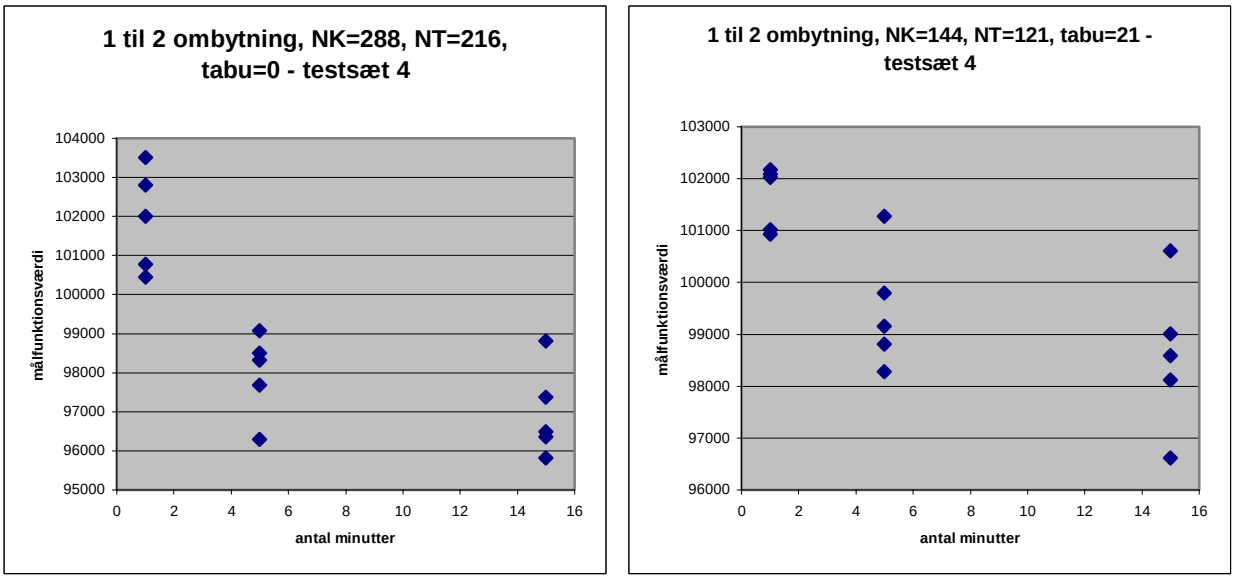
Betragtes ovenstående figur, bliver de bedste resultater opnået ved en tabuliste med længden 21 og NK=144, hvor der er henholdsvis 7, 6 og 9 bedste resultater ud af 10 mulige. Der kan laves mange flere tests for at underbygge eller finde bedre værdier, men grundet tidspres bliver værdierne for 1 til 1 ombytning nu låst til en tabuliste med længden 21 og NK=144.

Går vi over til 1 til 2 ombytning, kommer der udover tabulistens længde og NK også værdien for indsættelse af terminal NT.



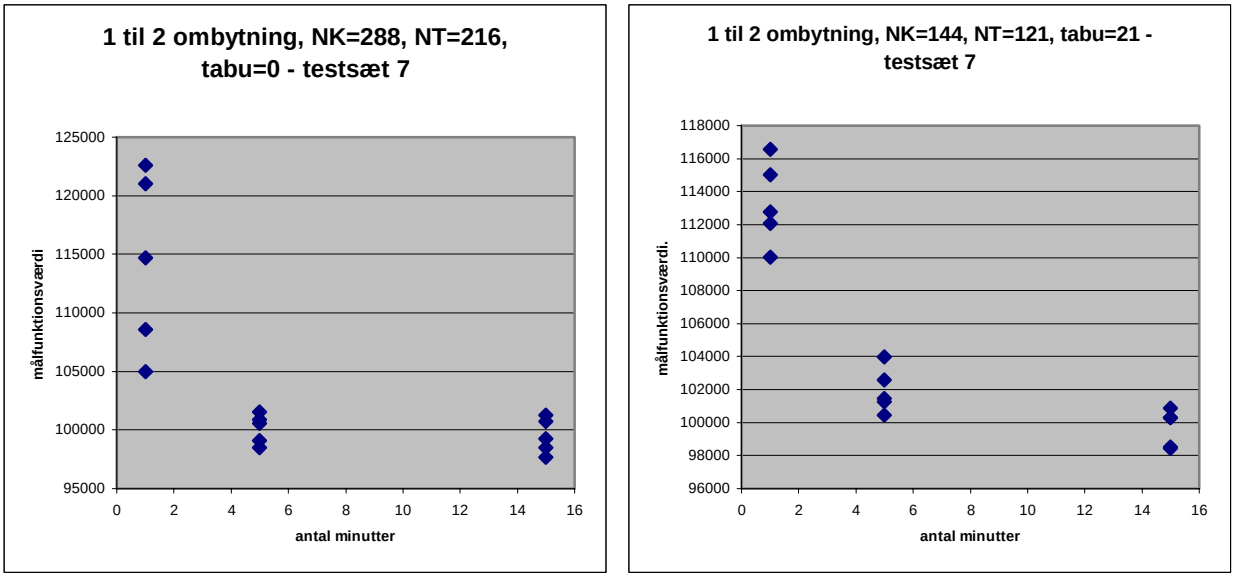
Figur 71 a og b – 1 til 2 ombytning – testsæt 1

Figur 71a har en dårlig start ved 1 minut, men både ved 5 og 15 minutter er der værdier under 80.000. På Figur 71b er starten lidt bedre, men især ved 5 minutter, men også ved 15 minutter, er b underlegen i forhold til Figur 71a.



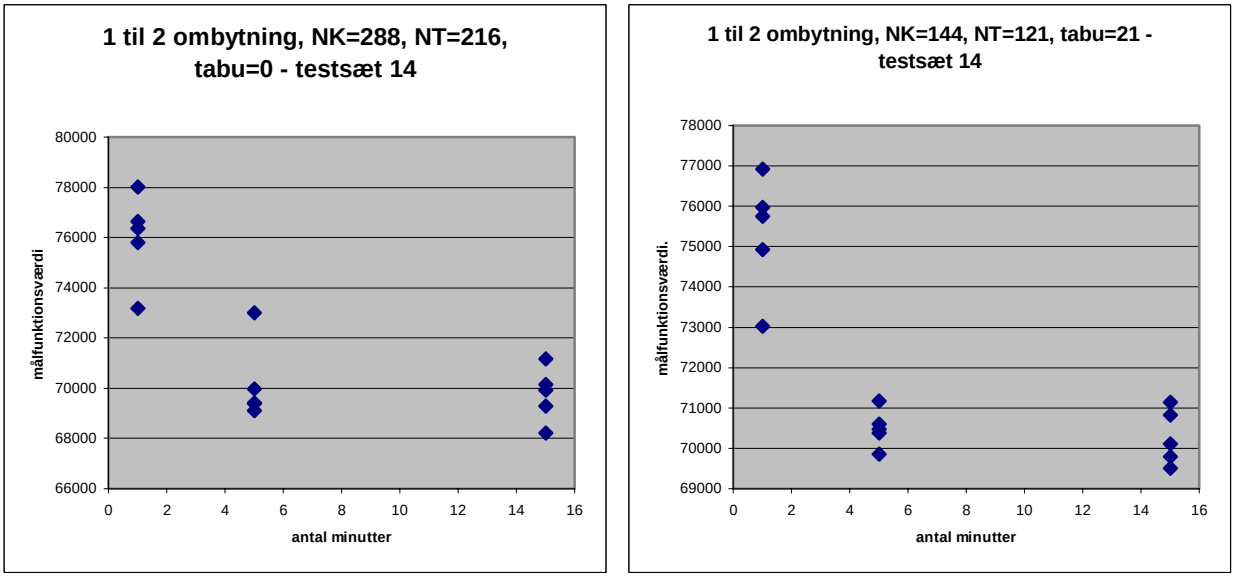
Figur 72 a og b – 1 til 2 ombytning – testsæt 4

Figur 72 udviser nogenlunde samme opførsel som Figur 71. Især ved de længere kørselstider er figur a overlegen i forhold til figur b.



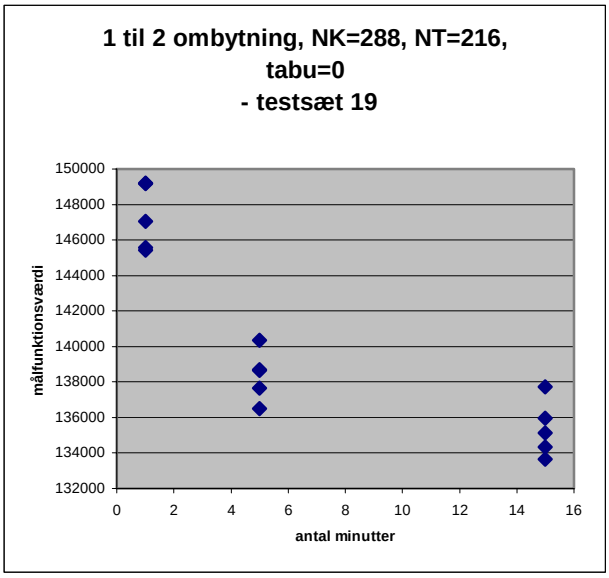
Figur 73 a og b – 1 til 2 ombytning – testsæt 7

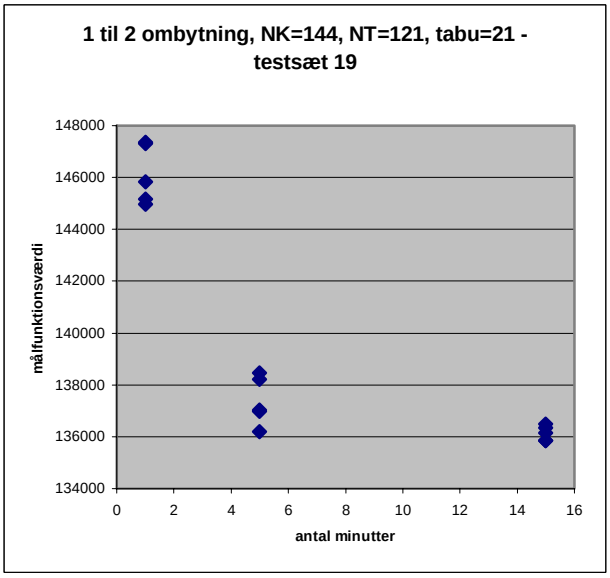
På Figur 73 snyder inddelingen af graferne lidt, men Figur 73a har henholdsvis 2 og 3 punkter under 100.000 for 5 og 15 minutter, mens Figur 73b kun har 1 punkt under 100.000 ved 15 minutter. Figur 73a har til gengæld utrolig stor spredning ved 1 minut.



Figur 74 a og b – 1 til 2 ombytning – testsæt 14

På Figur 74a lader det til. at løsningerne ved 5 minutter er mindst lige så gode. som løsningerne ved 15 minutter i gennemsnit. Heldigvis reddes de 15 minutter ved at have 1 punkt bedst placeret, men stadigt en uheldig tendens, der kunne skyldes at den nedre grænse var tæt på de 68.000. Figur 74b placerer sig lidt højere end Figur 74a men er alligevel godt med.





Figur 75 a og b – 1 til 2 ombytning – testsæt 19

I det sidste testsæt på Figur 75 er stillingen ved 1 og 5 minutter stort set uafgjort, men ved 15 minutter viser figur a de bedste løsninger. Den bedste løsning for a ligger et par tusinde under den bedste løsning fra b.

Kigger man nærmere på de bedste løsninger og gennemsnittet ved hver kørsel vha. Figur 76, er det klart, at den første 1 til 2 ombytning klarer sig klart bedst ved 5 og 15 minutters regnetid. Dette kan lettest konstateres ved at se på de pågældende søjler, der har de bedste resultater i henholdsvis 8 og 10 ud af 10 muligt. Derimod er den anden 1 til 2 ombytning, med en tabuliste med længden 21, overlegen ved 1 minuts kørselstid. Generelt vurderer jeg dog, at resultaterne ved 1 minuts kørselstid er mindre vigtige end resultaterne ved 5 og 15 minutter, da gevinsten ved at køre i 5 eller 15 minutter opvejer det øgede tidsforbrug. Dette afhænger selvfølgeligt af situationen. Det giver typisk en reduktion på mellem 5 og 10 procent i målfunktionsværdien, at tillade kørsler på 5 eller 15 minutter, i forhold til kørsler på kun 1 minut ifølge nedenstående figur.

datasæt	ombytning tid i minutter	1 til 2, NK=288, NT=216, tabuliste=0 1	5	15	1 til 1, NK=144, NT=121, tabuliste=21 1	5	15
1	bedste løsning	84959	**79679**	**79310**	**84254**	81380	79938
	gennemsnit	88909	**80654**	**80278**	**85688**	82102	80732
4	bedste løsning	**100450**	**96287**	**95816**	100928	98278	96613
	gennemsnit	101911	**97974**	**96969**	**101643**	99464	98588
7	bedste løsning	**104965**	**98495**	**97690**	110006	100451	98438
	gennemsnit	114361	**100130**	**99483**	**113283**	101946	99688
14	bedste løsning	73171	**69100**	**68212**	**73023**	69862	69510
	gennemsnit	75999	**70179**	**69745**	**75319**	70496	70276
19	bedste løsning	145431	136496	**133645**	**144969**	**136197**	135848
	gennemsnit	147284	138366	**135362**	**146130**	**137378**	136134
	antal markerede	2	**8**	**10**	**8**	2	0

Figur 76 – Sammenfatning af målfunktionsværdier for 1 til 2 ombytning med forskellige parametre

Samlet er den generelle tendens, at 1 til 2 ombytning klarer sig bedst med en tabuliste med længde 0, NK=288 og NT=216. Det bemærkes, at der kan laves mange flere tests, for at underbygge eller finde bedre værdier, men grundet tidspres bliver værdierne for 1 til 2 ombytning nu låst til de nævnte værdier.

Nedenstående figur viser en sammenligning mellem det bedste valg for 1 til 1 ombytning og 1 til 2 ombytning:

		1 til 2, NK=288, NT=216, tabuliste=0			1 til 1, NK=144, tabuliste=21		
datasæt	tid i minutter	1	5	15	1	5	15
1	bedste løsning	**84959**	**79679**	**79310**	91667	88291	87243
	gennemsnit	**88909**	**80654**	**80278**	92745	89217	87986
4	bedste løsning	**100450**	**96287**	**95816**	104915	102516	101073
	gennemsnit	**101911**	**97974**	**96969**	107658	103875	102412
7	bedste løsning	**104965**	**98495**	**97690**	114899	110926	109224
	gennemsnit	**114361**	**100130**	**99483**	118057	113396	110744
14	bedste løsning	**73171**	**69100**	**68212**	76436	73714	71324
	gennemsnit	**75999**	**70179**	**69745**	77055	74943	73103
19	bedste løsning	**145431**	**136496**	**133645**	154069	141705	139411
	gennemsnit	**147284**	**138366**	**135362**	154894	145122	140325
	antal markerede	**10**	**10**	**10**	0	0	0

Figur 77 – Sammenfatning af målfunktionerne for den bedste 1 til 2 ombytning og bedste 1 til 1 ombytning

Som forventet giver 1 til 2 ombytning bedre resultater i samtlige kørslen i forhold til 1 til 1 ombytning. Jeg vælger derfor kun at fortsætte med 1 til 2 ombytning med NK=288, NT=216 og en tabuliste med længden 0. Jeg ser ingen grund til at udregne resultater med andre ombytninger end den, der leverer de bedste resultater.

8 Resultater og Visualisering

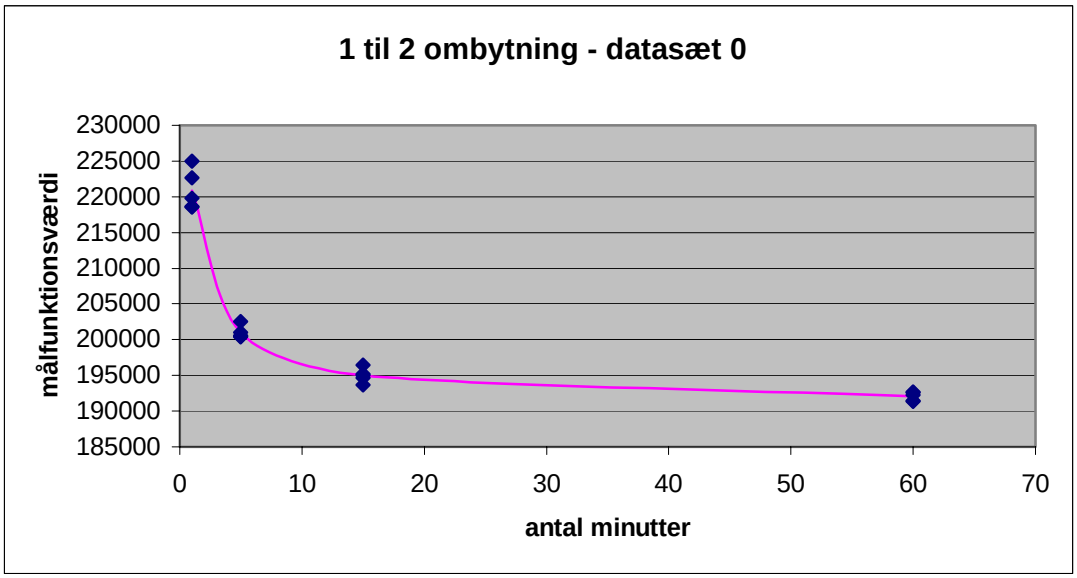
I dette afsnit vil jeg kigge på løsninger af de resterende 16 sæt, specielt det oprindelige datasæt, med 1 til 2 ombytning med fastlåste parametre.

Desuden vil jeg visualisere nogle af løsningerne vha. TRP og vise hvilke ændringer, der kan ske i en rute under et gennemløb af algoritmen.

8.1 Resultater

Jeg vil køre 5 tests ved 1 minut, 5 minutter, 15 minutter og desuden også 5 tests med 1 times varighed. Hver af de 20 kørsler per datasæt bliver plottet og desuden tegner jeg en linie igennem gennemsnittet ved hvert af de 4 tidspunkt, som du kan se på nedenstående figurer. Kørslerne ved 1 minut, 5 minutter og 15 minutter er, hvad jeg vurderer som en typisk kørsel i en anvendelsessituation af en algoritmen af denne art. Desuden er det også interessant med længere kørsler, hvilket jeg håber at tilfredsstille med 5 kørsler af 60 minutters varighed. I mangel på en optimal løsning, en anden muligt løsning eller en undergrænse jeg kan sammenligne med, er kørslerne ved 60 minutter eneste sammenligningsgrundlag for resultaterne fra de 3 andre tidspunkter.

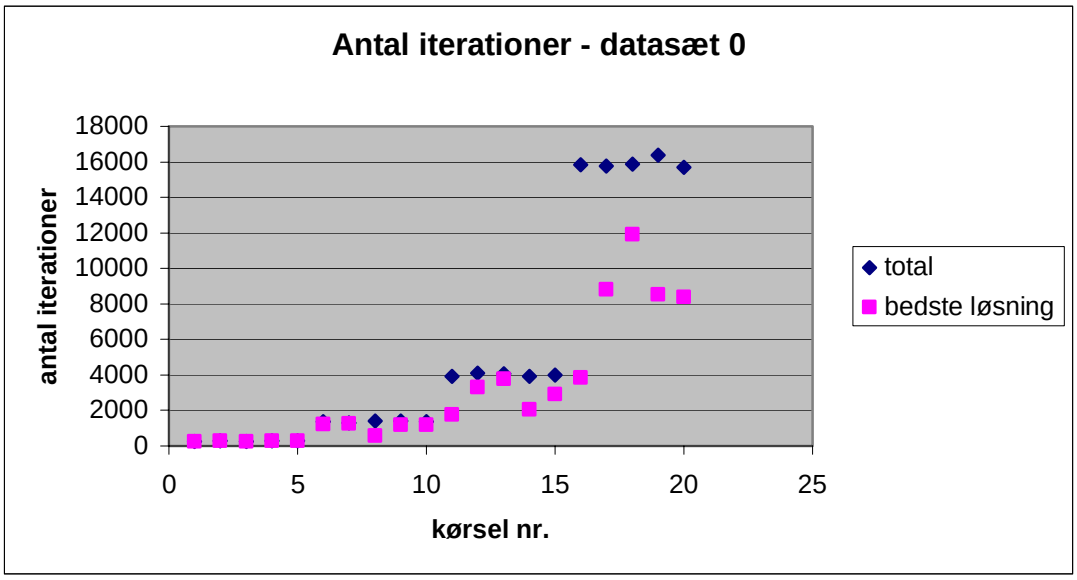
Som det første vil jeg gennemgå det oprindelige datasæt. Det har fået navnet datasæt 0. Det bliver testet ligesom de 15 resterende sæt.



Figur 78 – 1 til 2 ombytning – datasæt 0

Ovenstående figur viser løsningerne for det oprindelige datasæt med 4 terminaler, 60 vogne og 250 ordrer. Der er en betydelig spredning ved 1 minuts kørselstid, men spredningen bliver generelt mindre ved længere kørselstid og er meget lille ved 60 minutter. Kurven, der viser gennemsnittet ved hvert tidspunkt, er en pænt faldende kurve, præcist som man kunne forvente. Der ses stadigt en forbedring fra 15 minutter til 60 minutter på ca. 3000 i gennemsnit, men det betyder at målfunktionsværdien ved 15 minutter kun er 1,5% større end målfunktionsværdien ved 60 minutter.

På Figur 79 har jeg plottet det totale antal iterationer og antal iterationer for bedste løsning for hver kørsel knyttet til datasæt 0. Figuren er lavet på en lidt anderledes måde end Figur 78, der viser målfunktionsværdierne. På Figur 79 er hver kørsel vist i rækkefølge, da det ville være umuligt at skelne hvilke to punkter der hørte sammen, hvis jeg plottede alle kørsler med samme kørselstid oven i hinanden.

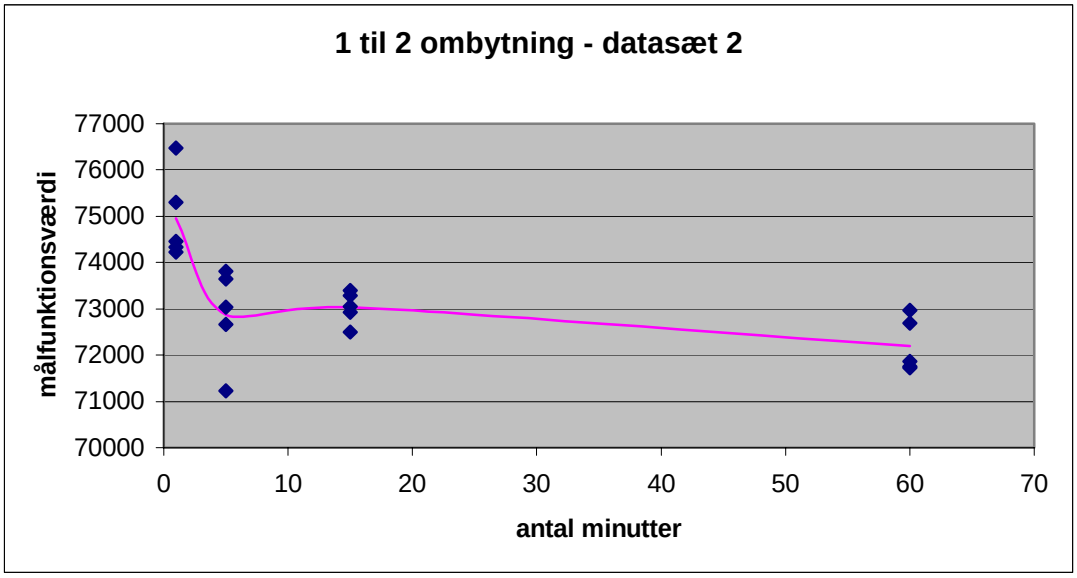


Figur 79 – Antal iterationer – datasæt 0

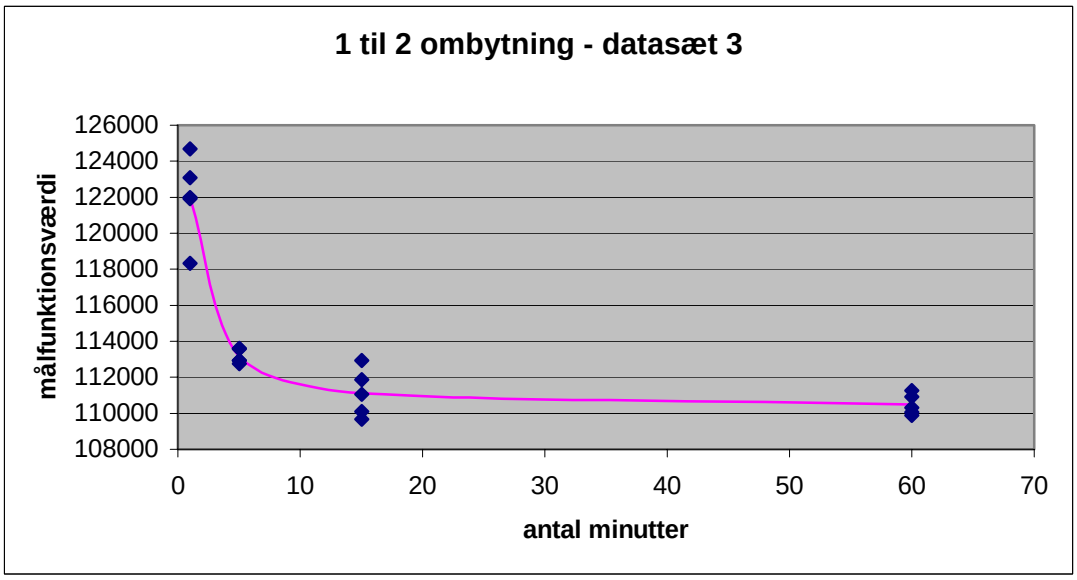
På ovenstående figur ses tydeligt, at antal iterationer for bedste løsning ligger meget tæt på det totale antal iterationer ved de første 5 kørsler af 1 minuts varighed. Dette tyder på at det er let at finde forbedringer efter så få iterationer. Kørsel 6 til 10 der er af 5 minutters varighed udviser samme opførsel, på nær kørsel 8, der ikke blev forbedret på de sidste iterationer. Ved 15 minutters kørselstid (kørsel 11 til 15) er der kun en kørsel, der finder sin bedste løsning tæt på dens totale antal iterationer. De øvrige løsninger ligger lidt under, men dog over niveauet for 5 minutter. Ved 60 minutters kørselstid er det tydeligt, at det er svært at finde forbedringer hele tiden, da flere af kørslerne ikke forbedrer deres bedste løsning i den sidste halvdel af kørslen. Kørsel 16 er lidt atypisk, da den finder en god målfunktionsværdi meget tidligt i forhold til de andre kørsler ved 60 minutter. Antallet af iterationer vises kun for nogle udvalgte sæt, da mange af figurerne er meget ens og ikke tilføjer noget nyt.

For sæt 2 nedenunder er det lidt andre resultater end for datasæt 0. Sæt 2 har på en eller anden måde fundet klart den bedste værdi i en af kørslerne ved 5 minutter. Noget tyder på at algoritmen virkeligt har været heldig i den ene kørsel eller at der er sket en fejl, for selv ved 60 minutter kommer den ikke tæt på resultatet igen. Det viser også noget om størrelsen af løsningsrummet, selvom sættet kun indeholder 2 terminaler, 31 vogne og 119 ordrer. Fjernes det ene afvigende punkt genetableres den forventede opførsel.

På Figur 80 skal man være opmærksom på, at inddelingen er forholdsvis lille, hvorfor resultaterne ser ud til at have en relativ stor spredning. Ser man bort fra det afvigende punkt ved 5 minutter, bliver kurven for gennemsnittet også en pæn kurve med den forventede opførsel.

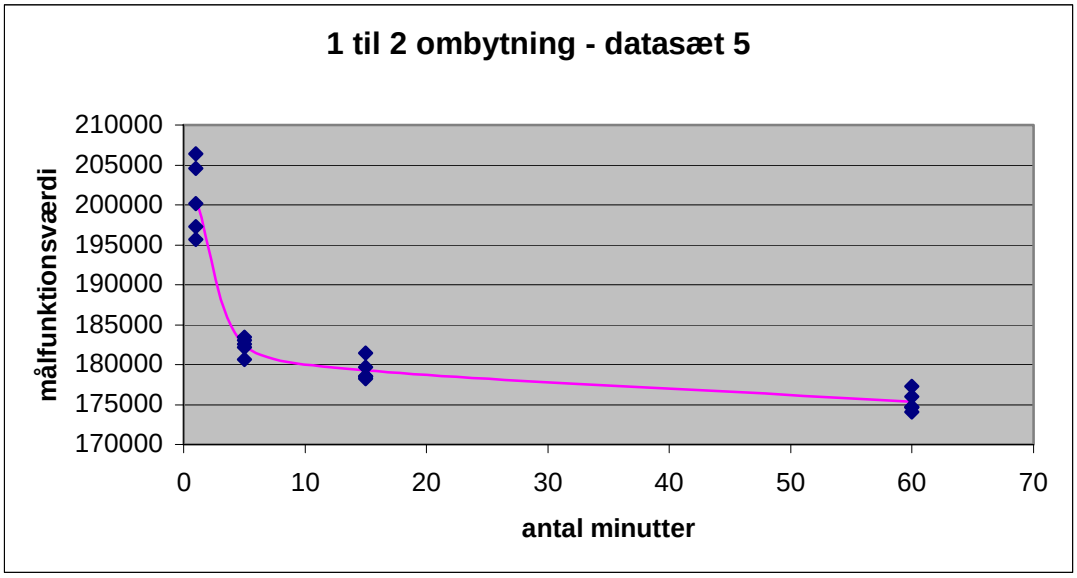


Figur 80 – 1 til 2 ombytning – datasæt 2



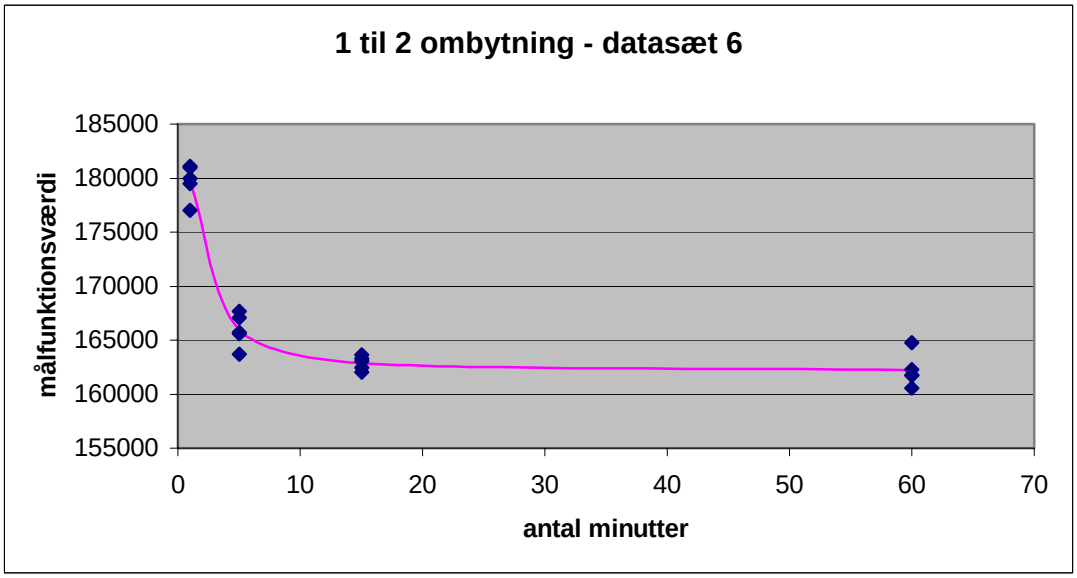
Figur 81 – 1 til 2 ombytning – datasæt 3

På Figur 81 er der en forholdsvis stor spredning ved 1 og 15 minutter, mens 5 og 60 minutter har en meget mindre spredning. I gennemsnit følger målfunktionsværdierne den forventede kurve, selvom den bedste løsning er fundet ved 15 minutter. Sættet har 151 ordrer fordelt på 26 vogne og 2 terminaler og forbedringen fra 15 til 60 minutter er på kun 0.6%.



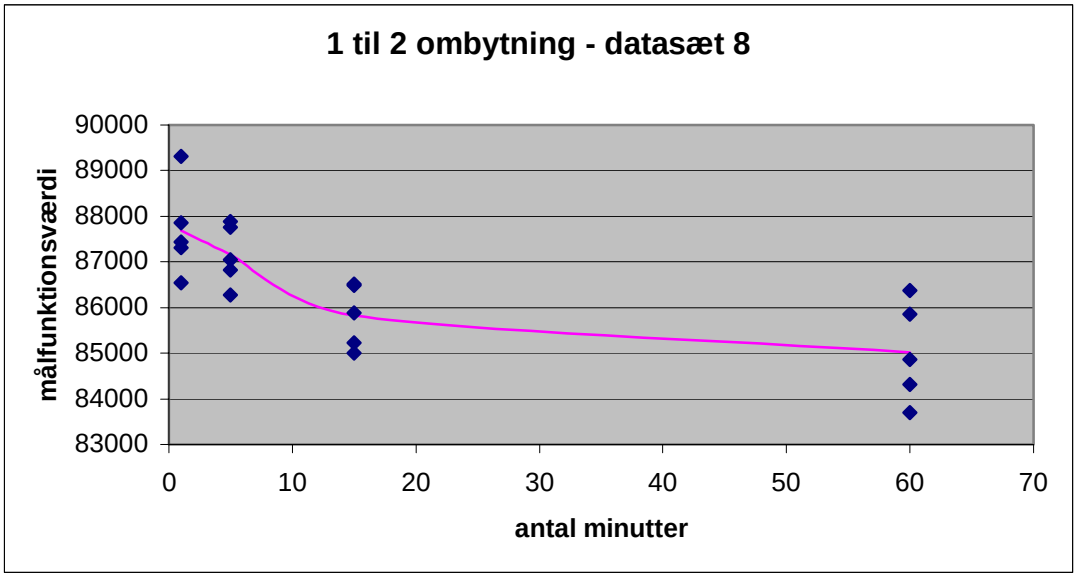
Figur 82 – 1 til 2 ombytning – datasæt 5

På Figur 82 ses resultaterne for sæt nr. 5. Igen ligger gennemsnittene på en pæn kurve, der starter brat, men flader ud hen mod de 60 minutter. Ved 60 minutter er resultaterne ca. 2.2% bedre end ved 15 minutter for det forholdsvist store sæt med 237 ordrer fordelt på 3 terminaler og 48 vogne. Allerede nu er det muligt at se, at de fleste figurer i dette afsnit minder om hinanden til en vis grad. Jeg vil derfor kun vise de figurer, der afviger fra den forventede opførsel, men på Figur 93 og Figur 94 er alle resultater sammenfattede i to tabeller.



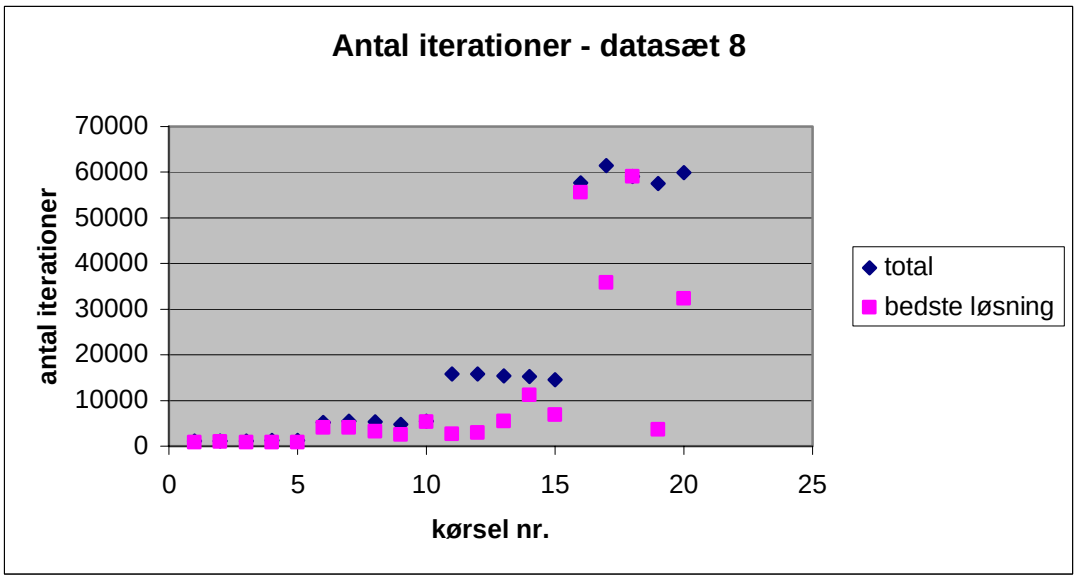
Figur 83 – 1 til 2 ombytning – datasæt 6

Det første jeg ser på Figur 83, er hvor fladt kurven for gennemsnittene er mellem 15 og 60 minutter. Det svarer til en forbedring på kun 0.4%. Der er desuden usædvanlig stor spredning ved de 60 minutter, med især et punkt der ligger dårligt. Det kan måske forklare den lille forbedring, for et ellers rimeligt stort sæt, med samme elementer som sæt 5, bare 226 ordrer i stedet for.



Figur 84 – 1 til 2 ombytning – datasæt 8

Sæt 8 har en usædvanlig opførsel med en noget atypisk kurve for de gennemsnitlige værdier, plus en stor spredning for alle resultaterne. Mest bemærkelsesværdigt er, at spredningen ved 60 minutter er den største af alle spredningerne for sættet. Den gennemsnitlige forbedring fra 15 til 60 minutter er 0.9% for sættet med 139 ordrer, 40 vogne og 3 terminaler. Forklaringen på de mærkelige resultater skal måske findes i figuren for antal iterationer nedenunder:

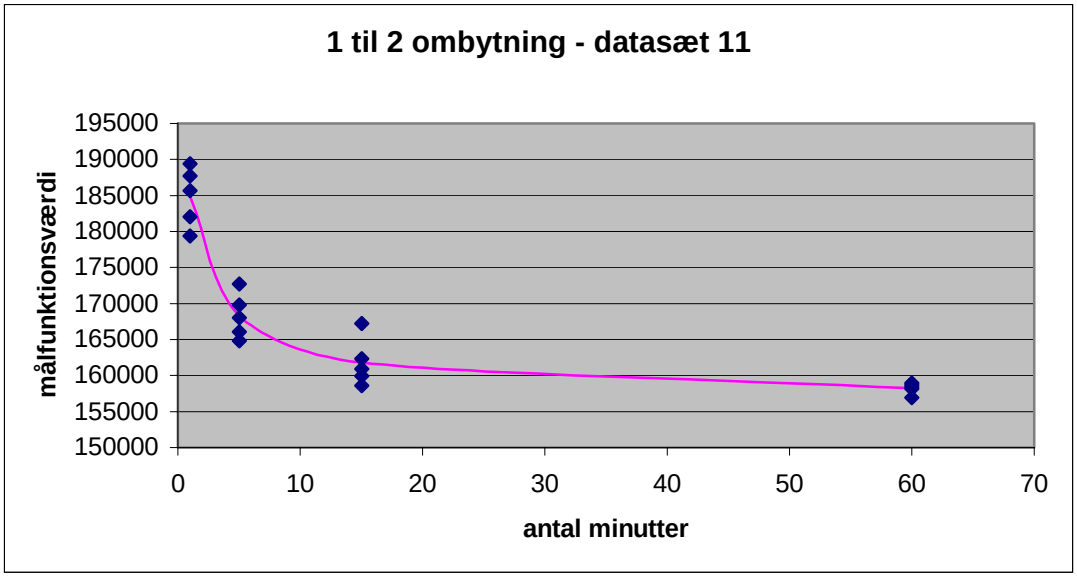


Figur 85 – Antal iterationer – datasæt 8

På Figur 85 ses en mængde bedste løsninger med meget lavt antal iterationer. Det totale antal iterationer er stabilt, men især ved 15 og 60 minutter går det galt for den bedste løsning. Hvorfor sæt 8 udviser denne opførsel, har jeg ikke en god forklaring på.

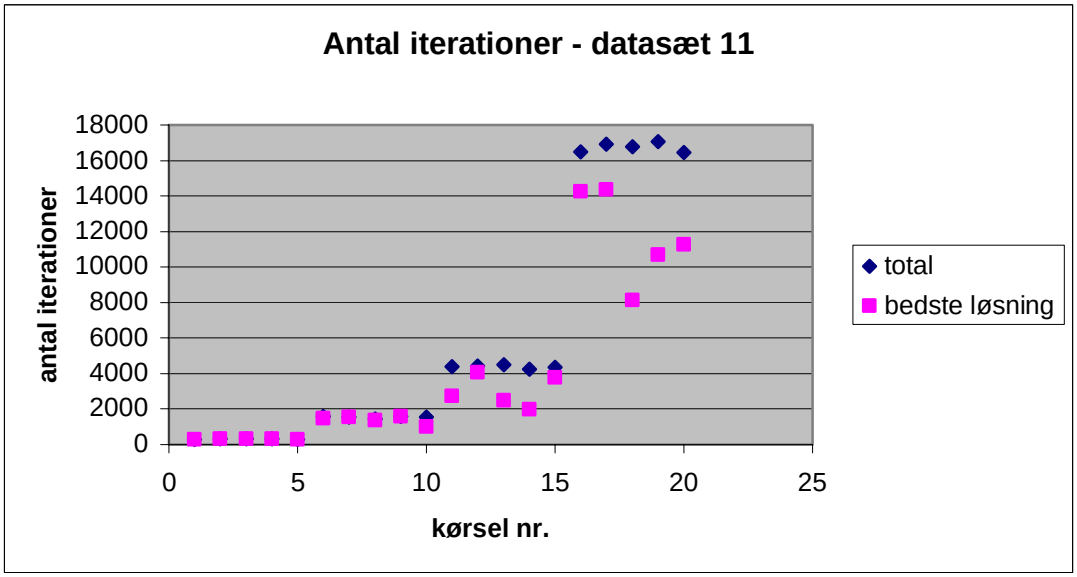
Under kørslerne for datasæt 9 opstod der flere gange fejl ved kald af oprettelse af ny rute. Det betød jeg skulle genstarte algoritmen manuelt og min computer derfor ikke kunne stå og køre stabilt i

løbet af natten osv. Jeg prøvede at finde fejlen, men det var ikke umiddelbart muligt på kort tid, så pga. tidspres så jeg ikke anden løsning end at slå denne feature fra.



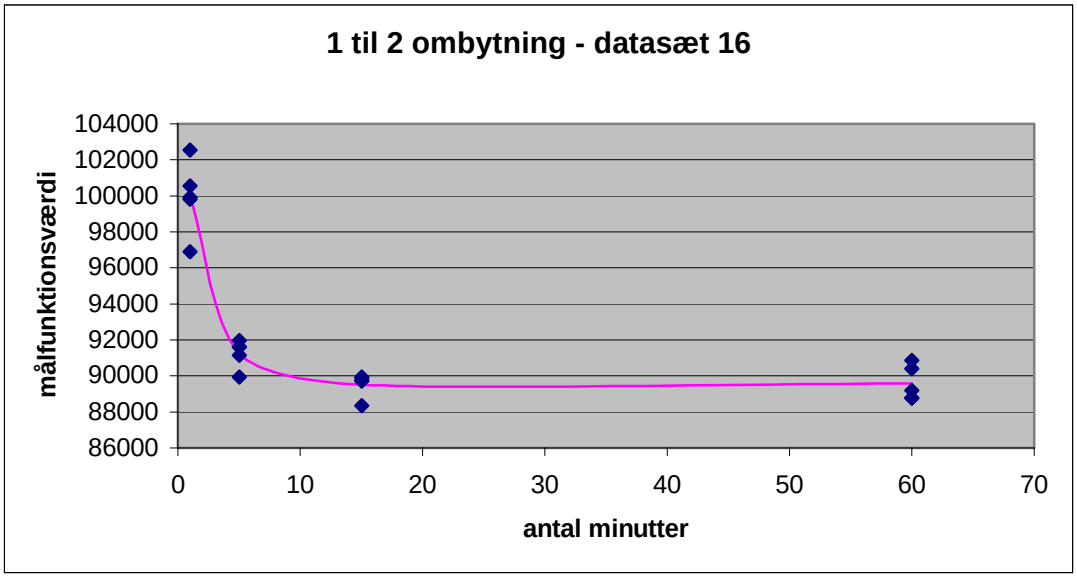
Figur 86 – 1 til 2 ombytning – datasæt 11

Sæt 11 har en afviger ved både 5 og 15 minutter, men et pænt resultat ved de 60 minutter. På figuren for antal iterationer nedenunder vil jeg se, om man kan skelne hvilken kørsel, der svarer til afvigerne.



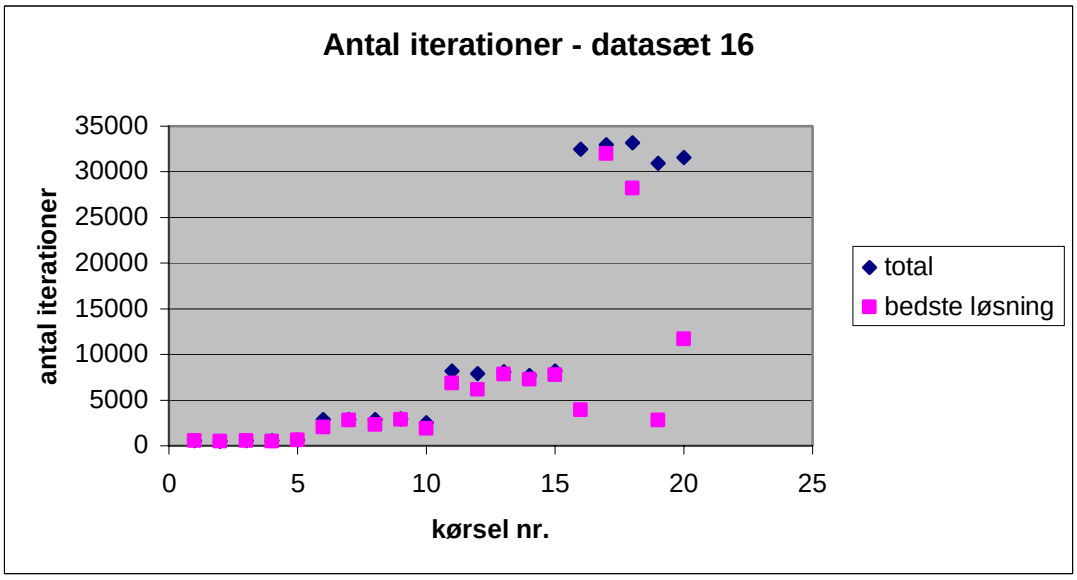
Figur 87 – Antal iterationer – datasæt 11

Den dårligste løsning ved henholdsvis 5 og 15 minutter svarer til kørsel nr. 10 og 13. Begge kørsler adskiller sig ikke betydeligt fra de andre for samme tidspunkt, faktisk giver kørsel 14, der har færre iterationer end kørsel 13, et rigtigt pænt resultat.



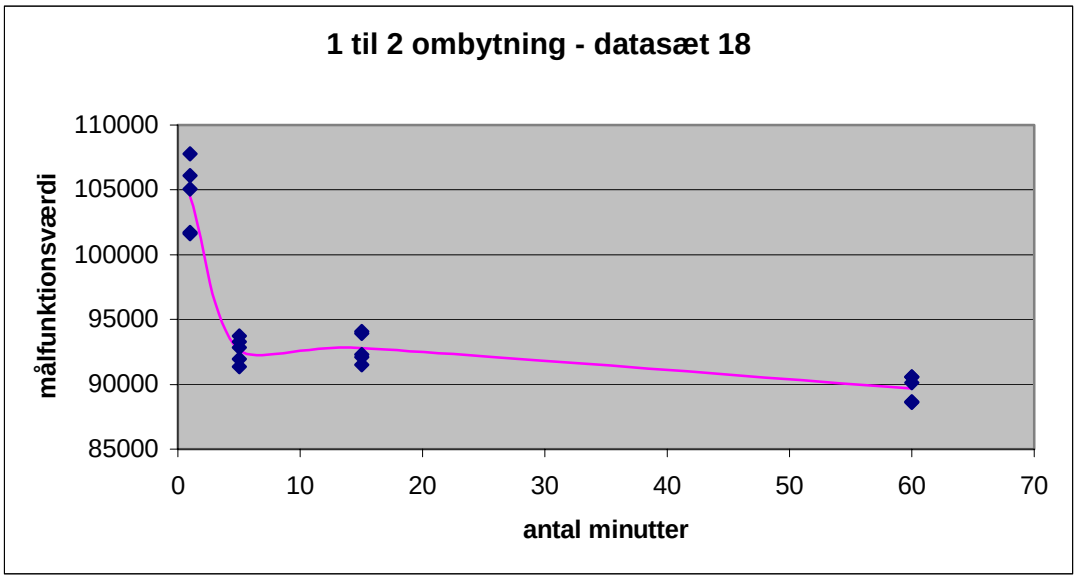
Figur 88 – 1 til 2 ombytning – datasæt 16

Sæt 16 er atypisk, da det som det eneste har en stigning i målfunktionsværdien i gennemsnit fra 15 til 60 minutter. Stigningen er ikke stor, men især 2 resultater ved de 60 minutter trækker meget op i gennemsnittet. Sættet indeholder 138 ordrer, kun 17 vogne og 2 terminaler. På næste figur er vist antal iterationer for sæt 16.



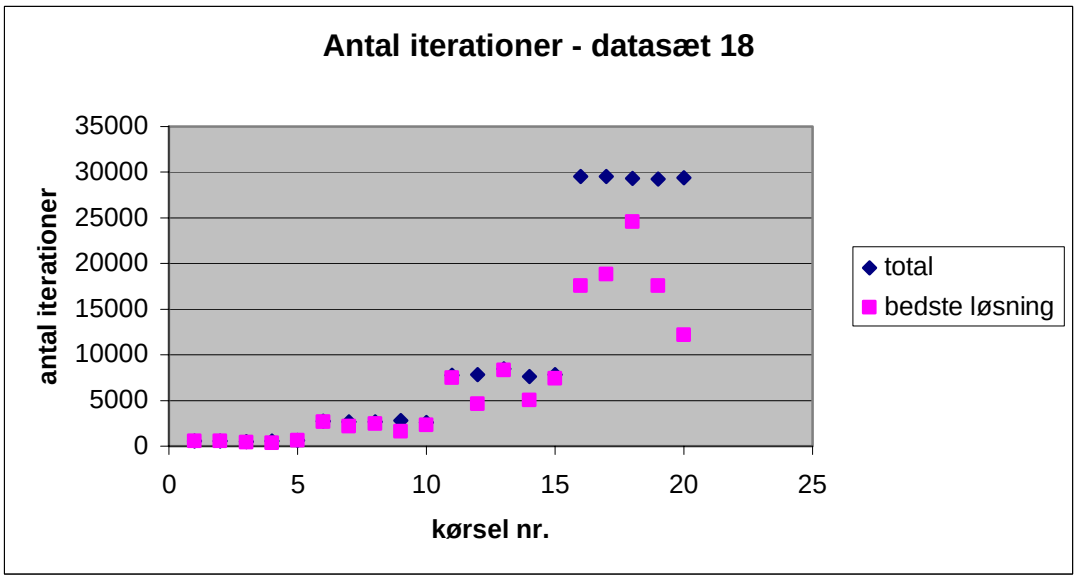
Figur 89 – Antal iterationer – datasæt 16

De to kørsler for 60 minutter, der opnår færrest iterationer for den bedste løsning (kørsel 16 og 19), er også de to kørsler, der giver den dårligste målfunktionsværdi. Det bemærkes, at begge kørsler opnår færre iterationer for den bedste løsning end alle 5 kørsler ved 15 minutter. Begge løsningers målfunktionsværdi er også dårligere end de 5 målfunktionsværdier ved 15 minutter.



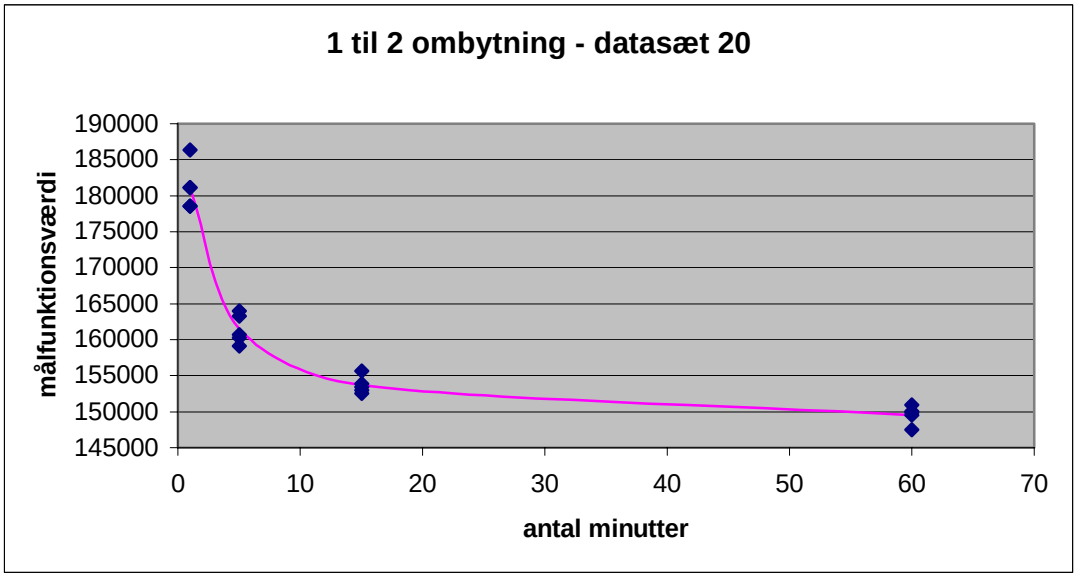
Figur 90 – 1 til 2 ombytning – datasæt 18

På Figur 90 er kurven for de gennemsnitlige målfunktionsværdier ikke som forventet. Det lader til at 2 punkter ved 15 minutter har opnået dårlige målfunktionsværdier. Nedenunder er antallet af iterationer for sæt 18.



Figur 91 – Antal iterationer – datasæt 18

De 2 løsninger med dårlige målfunktionsværdier ved 15 minutter svarer til kørsel nr. 12 og 13. Den dårligste løsning overhovedet ved 15 minutter er kørsel nr. 13, der ellers opnår det største antal iterationer. Igen ses at kørslerne ved 60 minutter ikke finder forbedrede løsninger blandt den sidste del af iterationerne, hvoraf jeg tolker at algoritmen er tæt på sin bedste værdi for sættet.



Figur 92 – 1 til 2 ombytning – datasæt 20

Sæt 20 på ovenstående figur følger det forventede mønster, selvom der er en forholdsvis stor forbedring fra 15 til 60 minutter på 2.7% i gennemsnit. Sættet er forholdsvis stort med 220 ordrer, 38 vogne og 3 terminaler, hvilket måske kan forklare, hvorfor der sker så relativt store forbedringer fra 15 til 60 minutter.

Opsummeres resultaterne i tabelform, er det muligt at danne sig et overblik via nedenstående figurer.

datasæt	tid i minutter	0	1	5	15	60
0	bedste løsning	283875	218549	200387	193696	191377
	gennemsnit	283875	220917	200995	194993	192069
2	bedste løsning	101093	74225	71224	72498	71721
	gennemsnit	101093	74958	72877	73033	72201
3	bedste løsning	161977	118324	112745	109686	109895
	gennemsnit	161977	121999	113158	111122	110489
5	bedste løsning	252777	195716	180652	178269	174116
	gennemsnit	252777	200826	182361	179286	175349
6	bedste løsning	226760	176989	163671	162050	160575
	gennemsnit	226760	179710	165941	162884	162230
8	bedste løsning	117396	86541	86281	84998	83705
	gennemsnit	117396	87694	87159	85821	85021
9	bedste løsning	215559	161749	152513	151241	149889
	gennemsnit	215559	164194	154129	152879	150962
10	bedste løsning	193784	142505	138676	135733	135055
	gennemsnit	193784	145563	139691	137354	136937
datasæt	tid i minutter	0	1	5	15	60
11	bedste løsning	231204	179397	164824	158644	156884
	gennemsnit	231204	184865	168289	161813	158248
12	bedste løsning	211925	165845	148345	144331	142145
	gennemsnit	211925	168681	150669	145365	143599
13	bedste løsning	103459	81033	77251	75980	75611
	gennemsnit	103459	82401	78016	76822	76112
15	bedste løsning	140458	100243	97235	94744	94021
	gennemsnit	140458	102533	98414	95598	95018
16	bedste løsning	136935	96914	89957	88355	88760
	gennemsnit	136935	99947	91265	89519	89597
17	bedste løsning	230532	181555	160360	158282	154783
	gennemsnit	230532	184960	163208	159499	155926
18	bedste løsning	120020	101613	91368	91522	88593
	gennemsnit	120020	104458	92642	92781	89702
20	bedste løsning	222913	178535	159153	152507	147476
	gennemsnit	222913	181154	161506	153710	149561

Figur 93 – Sammenfatning af målfunktionsværdier for 1 til 2 ombytning

Her er alle 16 sæt listet med resultaterne for deres bedste løsning og gennemsnittet til 5 tidspunkt. Tidspunktet 0 refererer til målfunktionsværdien for den initielle løsning. Skaleres disse resultater i forhold til den initielle løsnings målfunktionsværdi, får man en figur som på Figur 94 nedenfor.

datasæt	tid i minutter	0	1	5	15	60	%
0	bedste løsning	100	77.0	70.6	68.2	67.4	1.2
	gennemsnit	100	77.8	70.8	68.7	67.7	1.5
2	bedste løsning	100	73.4	70.5	71.7	70.9	1.1
	gennemsnit	100	74.1	72.1	72.2	71.4	1.1
3	bedste løsning	100	73.1	69.6	67.7	67.8	-0.2
	gennemsnit	100	75.3	69.9	68.6	68.2	0.6
5	bedste løsning	100	77.4	71.5	70.5	68.9	2.3
	gennemsnit	100	79.4	72.1	70.9	69.4	2.2
6	bedste løsning	100	78.1	72.2	71.5	70.8	0.9
	gennemsnit	100	79.3	73.2	71.8	71.5	0.4
8	bedste løsning	100	73.7	73.5	72.4	71.3	1.5
	gennemsnit	100	74.7	74.2	73.1	72.4	0.9
9	bedste løsning	100	75.0	70.8	70.2	69.5	0.9
	gennemsnit	100	76.2	71.5	70.9	70.0	1.3
10	bedste løsning	100	73.5	71.6	70.0	69.7	0.5
	gennemsnit	100	75.1	72.1	70.9	70.7	0.3
datasæt	tid i minutter	0	1	5	15	60	%
11	bedste løsning	100	77.6	71.3	68.6	67.9	1.1
	gennemsnit	100	80.0	72.8	70.0	68.4	2.2
12	bedste løsning	100	78.3	70.0	68.1	67.1	1.5
	gennemsnit	100	79.6	71.1	68.6	67.8	1.2
13	bedste løsning	100	78.3	74.7	73.4	73.1	0.5
	gennemsnit	100	79.6	75.4	74.3	73.6	0.9
15	bedste løsning	100	71.4	69.2	67.5	66.9	0.8
	gennemsnit	100	73.0	70.1	68.1	67.6	0.6
16	bedste løsning	100	70.8	65.7	64.5	64.8	-0.5
	gennemsnit	100	73.0	66.6	65.4	65.4	-0.1
17	bedste løsning	100	78.8	69.6	68.7	67.1	2.2
	gennemsnit	100	80.2	70.8	69.2	67.6	2.2
18	bedste løsning	100	84.7	76.1	76.3	73.8	3.2
	gennemsnit	100	87.0	77.2	77.3	74.7	3.3
20	bedste løsning	100	80.1	71.4	68.4	66.2	3.3
	gennemsnit	100	81.3	72.5	69.0	67.1	2.7
samlet	bedste løsning	100	76.3	71.1	69.9	69.0	1.3
	gennemsnit	100	77.9	72.0	70.6	69.6	1.4

Figur 94 – Sammenfatning af skalerede resultater for 1 til 2 ombytning

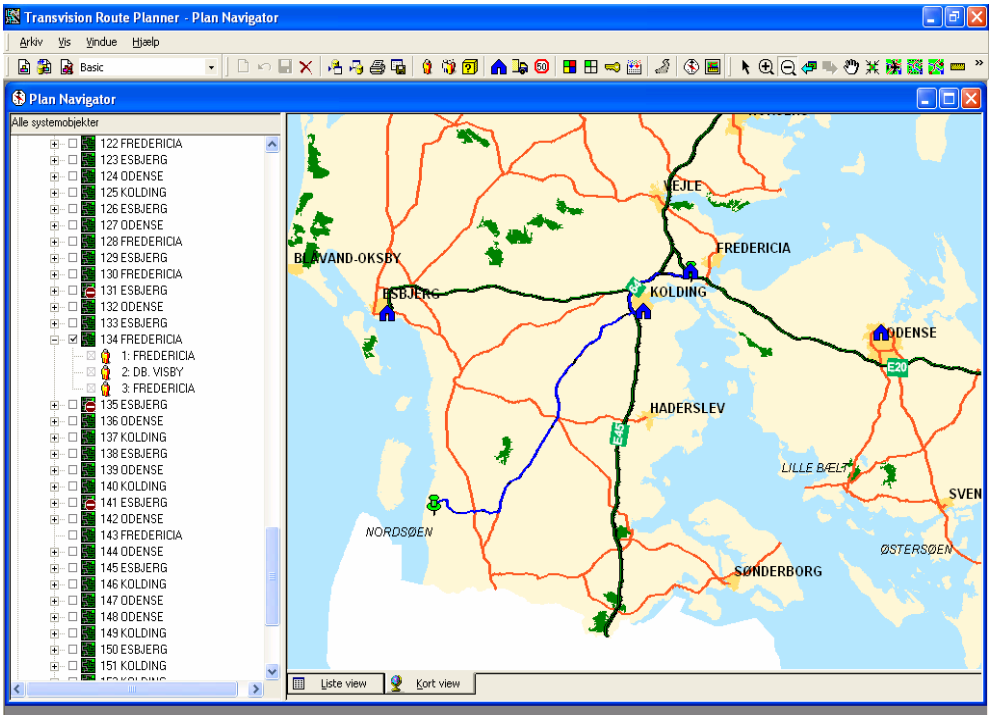
På ovenstående Figur 94 er alle resultater sammenfattet i en skaleret version. Udgangspunktet er den initielle løsnings målfunktionsværdi, der er sat til 100. For hvert sæt er det muligt at se forholdet mellem den initielle løsnings målfunktionsværdi og målfunktionsværdierne for hver af de tidsbestemte kørsler, både for den bedste løsning og i gennemsnit. I sidste søjle er opført den procentvise forbedring for målfunktionsværdierne ved at køre 60 minutter i forhold til 15 minutter. Denne procentsats er et af de vigtigste redskaber til at bedømme hvor godt algoritmen klarer sig.

Specielt sæt 16 er besynderligt, da det som det eneste sæt i gennemsnit har en dårligere løsning ved 60 minutter end ved 15 minutter. Alle andre sæt har i gennemsnit de bedste målfunktionsværdier ved 60 minutter. Samlet for alle sættene gælder, at den bedste løsning bliver forbedret 1.3% ved at lade algoritmen køre 60 minutter i stedet for kun 15 minutter, mens den gennemsnitlige løsning forbedres 1.4%.

8.2 Visualisering

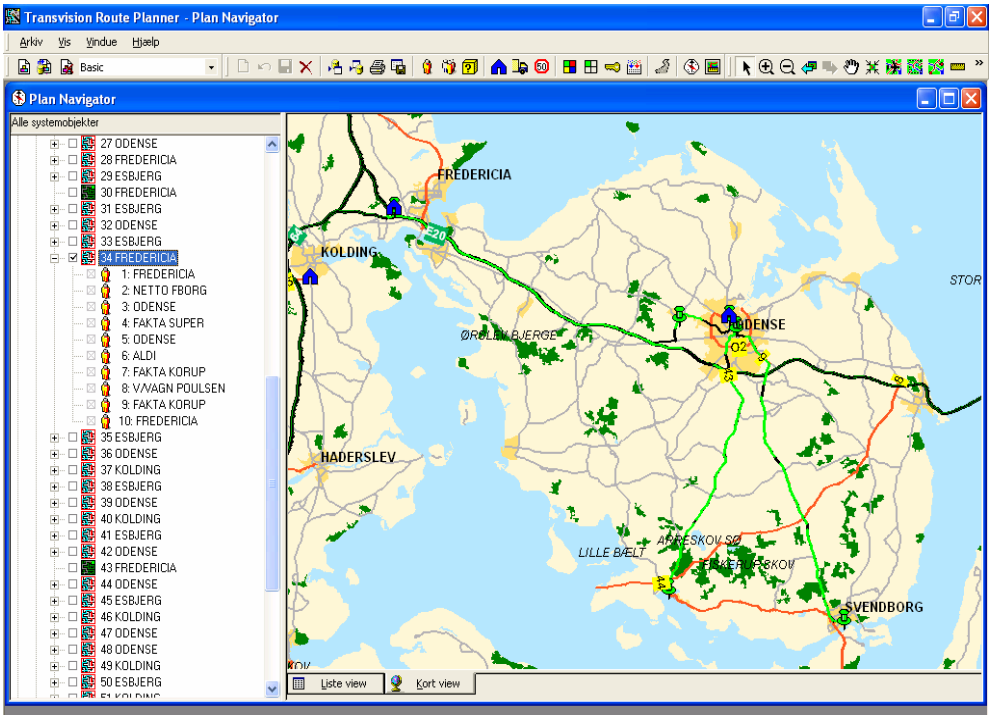
I dette afsnit vil jeg vha. TRP demonstrere nogle af løsningerne til datasæt 0, både fra den initielle løsning og fra den bedst fundne løsning.

Nedenunder ses rute 34 fra den initielle løsning. Ruten starter i terminalen i Fredericia, kører til Visby og tilbage til Fredericia. Vognen der kører rute 34 har både overskydende kapacitet og tid.



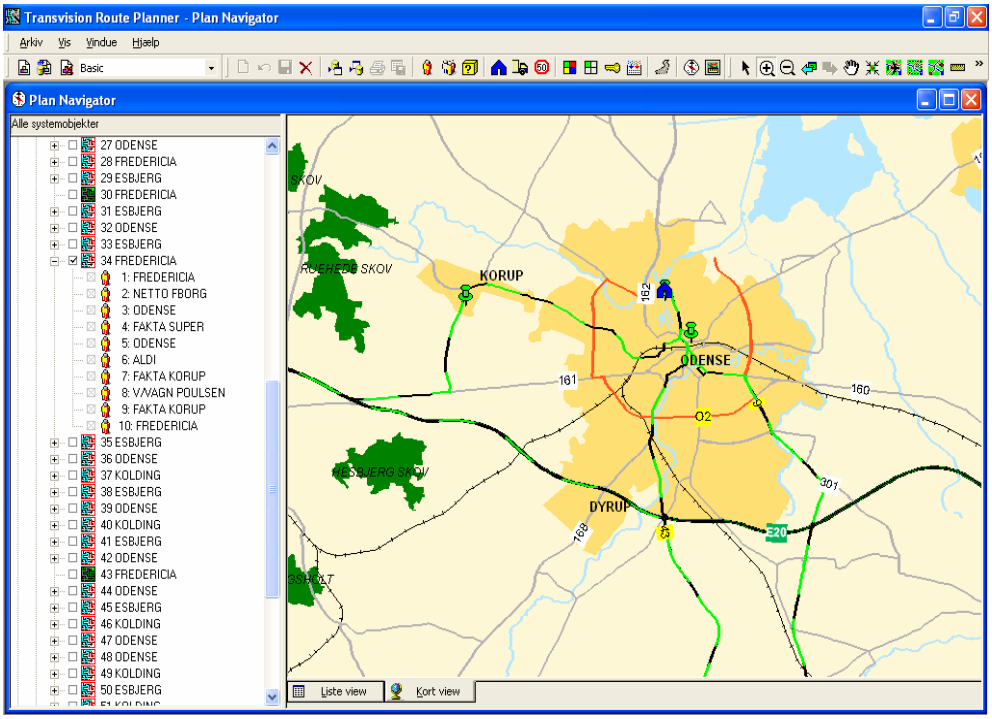
Figur 95 – Rute 34 fra den initielle løsning visualiseret i TRP

I den bedste løsning bliver rute 34 ændret fuldstændigt og vognen laver to stop ved terminalen i Odense udover stoppene ved start- og slutterminalen i Fredericia. Vognens kapacitet og tid bliver udnyttet i højere grad end i den initielle løsning.



Figur 96 – Rute 34 fra den bedste løsning visualiseret i TRP

Zoomer jeg ind på kortet i TRP, er det tydeligere at se hvordan vognen kører rundt i Odense.

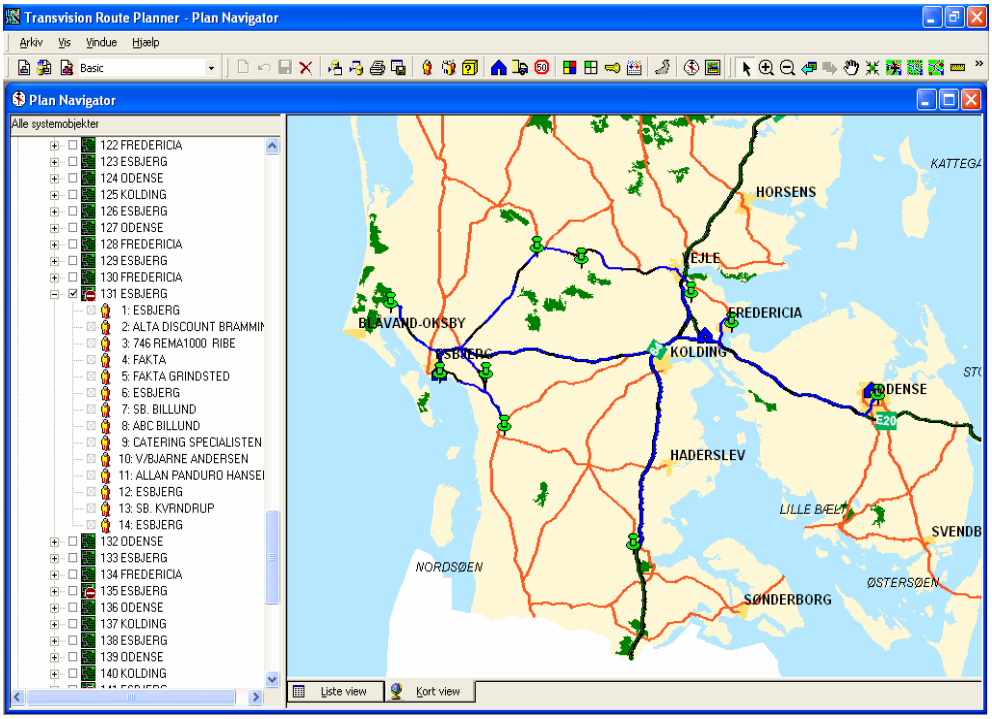


Figur 97 – Rute 34 fra den bedste løsning zoomed ind på Odense

Vognen 34 kommer ind fra venstre med last fra Fredericia, kører gennem Dyrup og videre af vej 43 ned til Netto Fåborg. Efter at aflæsningen i Fåborg køres til terminalen i Odense, hvor der hentes last til ordren fra Fakta Super i Svendborg via vej 9. Efter turen i Svendborg returnerer vognen til

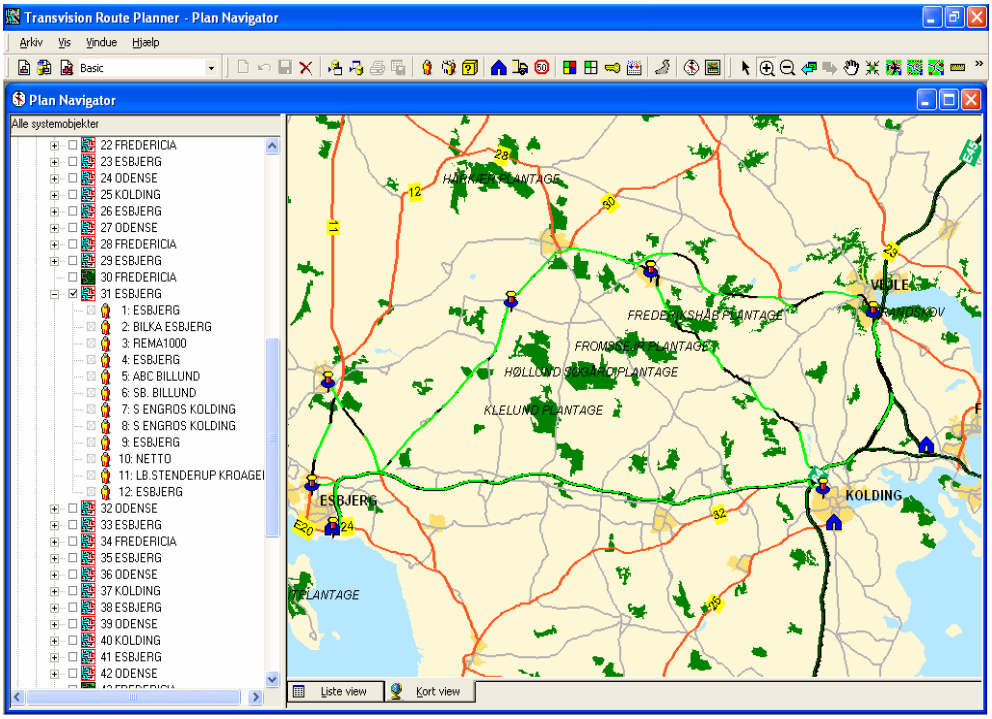
Odense, hvor der opsamles last til de sidste 4 ordrer. Den første af disse ordrer er Aldi i Odense og de sidste 3 ligger ude i Korup. Til slut returnerer vognen til Fredericia.

Ruterne i den initielle løsning kan også være meget store. F.eks. rute 31, der starter og slutter i Esbjerg, men kommer forbi både Kolding, Fredericia og Odense. Vogn 31 må dog ikke stoppe i disse byer i den initielle løsning.



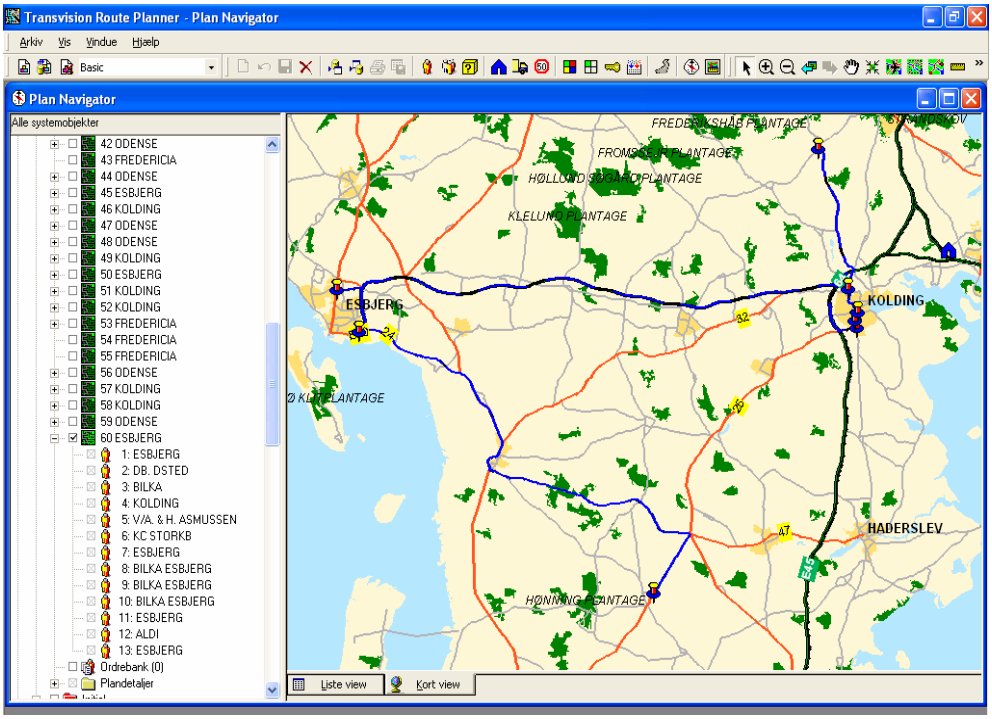
Figur 98 – Rute 31 fra den initielle løsning

I den bedste løsning er rute 31 ændret til følgende:



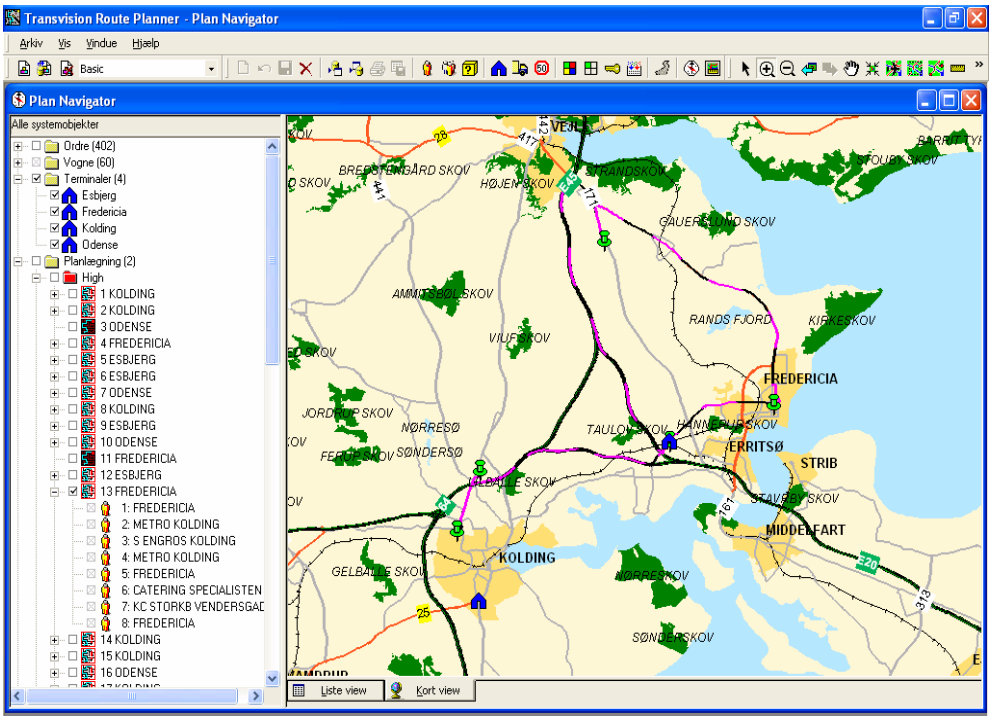
Figur 99 – Rute 31 fra den bedste løsning

Rute 31 benytter stadigt kun terminalen i Esbjerg, men har begrænset sig til at servicere ordrer nord for Esbjerg og Kolding. Desuden er det værd at bemærke, at to ordrer serviceres i Kolding, selvom vognen ikke stopper der, hvilket en menneskelig planlægger måske ikke ville tro var smart.



Figur 100 – Rute 60 fra den bedste løsning

På ovenstående figur er rute 60 visualiseret. Igen servicerer en vogn fra Esbjerg ordrer i Kolding, denne gang stopper vognen dog ved terminalen i Kolding.



Figur 101 – Rute 13 fra den bedste løsning

På Figur 101 kan man se, at Metro Kolding serviceres to gange men ikke lige i træk. Ordre 2 og 4 er det nordlige punkt i Kolding, mens ordre 3 er det sydlige punkt i Kolding. Da Metro Kolding ligger et stykke fra vejen mellem ordre 3 og terminalen i Fredericia, kører vogn 13 en omvej ved at tage ordrerne i den rækkefølge. Det er dog den eneste ting jeg har fundet, hvor man umiddelbart kan se at løsningen ikke er den bedste mulige. Den type fejl er dog både let at opdage og tage højde for, selvom jeg ikke har gjort det i dette projekt.

9 Konklusion

Der er flere måder at teste om en løsning fra en heuristik er god. Alle måder jeg kender er baseret på en sammenligning mellem løsningen fra heuristikken og en anden løsning, der kan fremkomme på en af følgende måder:

- Find en optimal løsning vha. en matematisk model.
    - Denne måde er den mest præcise, da den optimale løsning giver det bedst opnåelige resultat. Det er til gengæld ofte svært at lave en matematisk model, der kan løses for den optimale løsning.
- Find en nedre grænse for problemet vha. en relaxering af problemet.
    - I forhold til den optimale løsning er det lettere at lave en relaxering af problemet. Det giver bare ikke den optimale løsning, ej heller en løsning man kan være sikker på er god. Men en nedre grænse giver dog et maksimum tal for, hvor langt man er fra den optimale løsning, da denne ligger et sted mellem den fundne værdi og den nedre grænse.
- Find løsninger fra andre metoder, f.eks. andre heuristikker, for samme problem.
    - Udføres et optimeringsarbejde for en virksomhed kan det være muligt at sammenligne med deres tidligere løsninger af tilsvarende problemer. Sammenligningen vil vise om optimeringen er mere effektiv end deres tidligere måder, hvilket tydeligvis er et succeskriterium i den situation. Til gengæld siger det intet om forskellen mellem de fundne løsninger og den optimale løsning.

Som det fremgår af denne rapport, var det ikke muligt at finde en optimal løsning via en matematisk model og heller ikke muligt at lave en fornuftig relaxering, der ville give et brugbart resultat.

Derimod sammenligner jeg resultaterne opnået ved 1, 5 og specielt 15 minutter i forhold til resultaterne ved 60 minutters kørselstid. Kørselstiderne på 1, 5 og 15 minutter er den anslåede kørselstid i en anvendelsessituation i virkeligheden. Kørslen på 60 minutter giver en markant bedre gennemgang af løsningsrummet og tjener som reference til løsningerne opnået ved 1, 5 og 15 minutter. Samlet for alle 16 sæt testet i kapitel 8 gælder, at den bedste løsning bliver forbedret 1.3% og den gennemsnitlige løsning 1.4% ved at køre 60 minutter i stedet for kun 15 minutter.

Selvom det ikke var muligt at finde en optimal løsning eller en nedre grænse, er målene i opgaven opfyldt.

- Der er lavet en løsning på produktions- og distributionsproblemet både i det generelle og det specifikke tilfælde.
    - Herunder er der genereret planer for hver enkelt vogns rute og en plan for hver terminals produktionsmængde. Disse ruter er kompatible med Transvision Route Planner (TRP) og kan importeres og visualiseres i TRP.
- Løsningerne, der er fundet, er en minimering af omkostningerne indenfor få procent af den bedste kendte værdi. Man må generelt anse resultater indenfor 2% af den bedste løsning som flot.

Mulige forbedringer
I et eksperimentelt projekt som dette, kan jeg finde mange alternative metoder der kunne afprøves og mulige forbedringer:

En bedre afprøvning og indstilling af parametrene der indgår i algoritmen. Jeg har ikke overblik over hvor stor en forbedring en bedre afprøvning og indstilling af parameter kunne give, men en yderligere dokumentation af dette ville i sig selv være værdifuldt.

En anden type tabuliste, hvor det ikke er overgange men selve ordren eller ruten man laver tabu. Dette går dog i modsat retning af artiklerne jeg har læst indenfor MDVRP, men kunne måske gøre tabulisten til en vigtig brik i 1 til 2 ombytning.

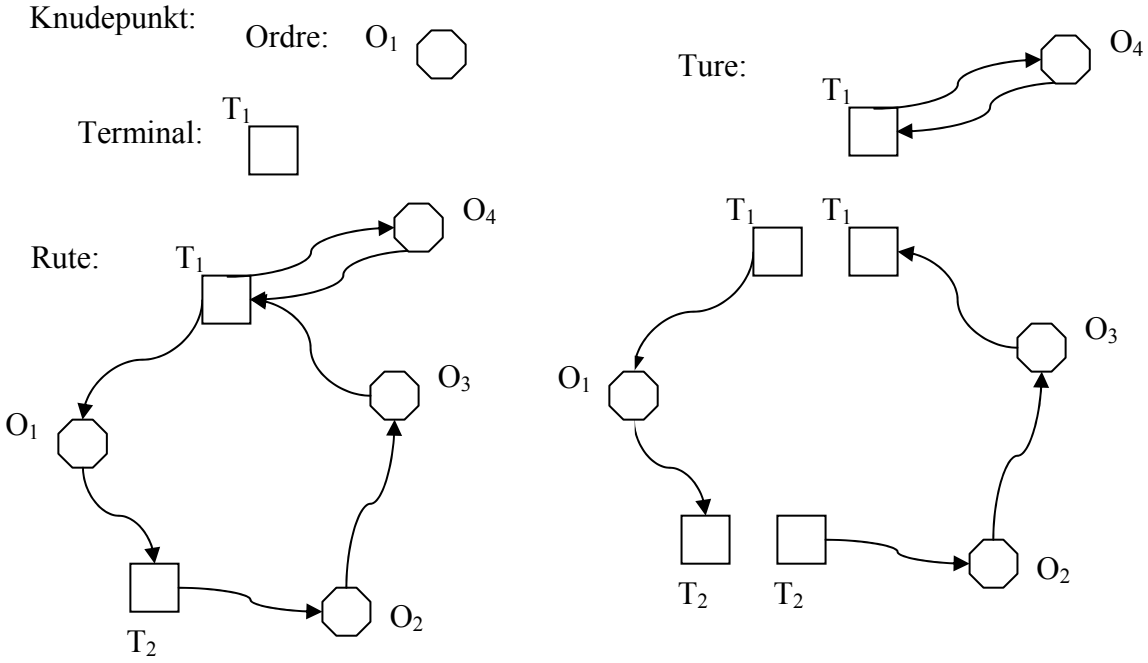
Eftersom det lader til, at mange korte søgninger kan give et bedre resultat end en lang søgning, kunne man forstille sig at starte med en løsning og køre x iterationer frem. Hvis ingen global forbedring er sket, springes tilbage til den oprindelige løsning og en ny søgning x skridt frem foretages. Det kan kombinere mange korte søgninger til en lang.

Desuden kunne der uden tvivl forbedres en del med hensyn til kørselstid af algoritmen ved at fokusere mere på hastighedsmæssige aspekter i programmeringen. Dette ligger udenfor denne opgave, men er nok det område, der skulle fokuseres mest på hvis mit program skulle benyttes i en virksomhed.

10 Appendiks

10.1 Notation

Knudepunkt	Fællesbetegnelse for en ordre eller en terminal
Tur	Mængde af knudepunkter der betjenes af en bestemt vogn mellem 2 terminalbesøg
Rute	Mængden af alle ture en bestemt vogn kører



Figur 102 – Illustration af ordre, terminal, knudepunkt, ture og ruter

Nabolag	Fra engelsk neighborhood, også kendt som omegn i dansk faglitteratur. Beskriver et løsningsrum.

10.2 Data

På vedlagte CD-ROM er henholdsvis de 20 testsæt og det oprindelige datasæt, kildekoden til C# programmet og udvalgte Excel-regneark benyttet i rapporten.
De 20 testsæt og det oprindelige datasæt er placeret i mapper med samme nr. som på Figur 40.

11 Bibliografi

[Chu98] P.C.Chu og J.E.Beasley, ”Constraint Handling in Genetic Algorithms: The Set Partitioning Problem”, Journal of Heuristics, 11 (1998) 323–357

[Cor01] J-F Cordeau, G Laporte og A Mercier, “A unified tabu search heuristic for vehicle routing problems with time windows”, Journal of the Operational Research Society 52 (2001) 928-936

[Glo95] Fred Glover, James P. Kelly og Manuel Laguna, ”Genetic Algorithms and Tabu Search: Hybrids for Optimization”, Computers and Operations Research vol 22 no 1 (1995) 111-134

[Lau94] P.S.Laursen, ”Generelle Optimeringsheuristikker – en introduktion”, Datalogisk Institut, Københavns Universitet (DIKU) (1994) 15-23

[Lap84] G. Laporte, Y. Nobert og D. Arpin, ”Optimal solutions to capacitated multidepot vehicle routing problems”, Congressus Numerantium 44, (1984) 283-292

[Lap88] G. Laporte, Y. Nobert og S. Taillefer, ”Solving a family of multi-depot vehicle routing and location-routing problems”, Transportation Science 22, (1988) 161-172

[Osm93] I. H. Osman, “Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem”, Annuls Operation Research 41, (1993) 421-451

[Ren96] Jacques Renaud, Gilbert Laporte og Fayez F. Boctor, ”A Tabu Search Heuristic for the Multi-Depot Vehicle Routing Problem”, Computers and Operations Research vol 23 no 3 (1996) 229-235

[Røp02] Stefan Røpke, “The Mulit-Vehicle Pickup and Delivery Problem with Time Windows”, DTU (2002).

[Sum95] Robert T. Sumichrast and Ina S. Markham, “A Heuristic and Lower Bound for a Multi-Depot Routing Problem”, Computers and Operations Research vol 22 no 10 (1995) 1047-1056

[You01] Habib Youssef, “Evolutionary algorithms, simulated annealing and tabu search: a comparative study”, Engineering Applications of Artificial Intelligence vol. 14 (2001) 167-181.