Projekt 3 i TAOP18 *Supply chain optimization*

Pedersen, Anton
antpe759@student.liu.se

Lindholm, David
davli921@student.liu.se

18 December 2016

Innehåll

1 Introduktion

Denna uppgift går ut på att personalplanera för Air New Zealands flygningar med hembasen belägen i Auckland. Enligt uppgiftsbeskrivningen är personalkostnader den andra största kostnadsdrivaren för ett flygbolag. Flygmarknaden går mer mot lågpris, ett tydligt exempel på det är att SAS nyligen meddelade att de överväger att etablera baser utanför skandinavien för att undkomma höga personalkostnaderna. Det känns därför spännande och relevant att lösa uppgiften eftersom att en bra planering är i det närmaste en hygienfaktor för att Air New Zealand fortsättningsvis ska vara konkurrenskraftiga på en allt tuffare marknad.

Den här uppgiften fokuserar på att hitta en kostnadseffektiv sekvens av flygninar för flygpersonalen, problemet kallas the pairing problem. I *the pairing problem* konstrueras giltiga sekvenser av flygningar (som kallas för Tours of Duty). Efter att en giltiga Tours of Duty (ToDs) skapas ska *the rostering problem* lösas vilket går ut på att tilldela flygpersonal en ToD.

2 Modellformulering

2.1 Introduktion

För att lösa problemet ska följande matematiska modell lösas

- Låt $x_j = 1$ ifall ToD j används och 0 annars.
- $a_{ij} = 1$ ifall flyg i ingår i rutt j 0 annars.
- c_j är kostnaden för att flyga ToD j.

$$\min z = \sum_j c_j \cdot x_j$$

då

$$\sum_j a_{ij} \cdot x_j \geq 1 \qquad \forall i \tag{1}$$

$$x_j \in \{0,1\} \qquad \forall j$$

Tolkningen av (1) är att alla flygningar ska ingå i minst en ToD. Man inser snabbt att problemet blir extremt stort, eftersom att antalet rutter som kan skapas är väldigt många. Det skapar två problem, dels kommer a-matrisen bli enormt stor och dels problemet kommer få enormt många variabler eftersom att j (det vill säga antalet ToD) kommer att bli enormt stor. För att undvika den problematiken används kolumngenerering. Problemet delas nu in i två problem: ett masterproblem och ett subproblem.

Vårt masterproblem är av typen övertäckningsproblem (eng. Set Covering Problem) och vårt subproblem är ett problem av typen kortaste väg problem (eng.

Shortest Path Problem). Vi använder oss utav kolumngenering för att succesivt generera fram lösningar som har en reducerad kostnad som är mindre än noll, det vill säga lösningar som är givande för målfunktionen att använda sig av. Vi gör kolumngenereringen för ett LP-relaxerat problem av vårt problem. Om vi har hittat en lösning som är större än noll så stoppar vi vår generering och har då hittat LP-optimum.

Branch and bound används för att hitta en heltalslösning, det vill säga en heltalslösning kommer att hittas med de ToDs som har genererats fram i kolumngenereringen. Istället för att använda sig av branch and bound så skulle man kunna använda sig av branch and price, vilket är en variation på branch and bound där kolumngenerering görs i varje branch". Med denna metod skulle man garenterat kunna hitta heltalsoptimum, vilket med vår metod inte kan garanteras, det är troligt att vi inte kommer hitta heltalsoptimum.

Om problemet har heltalsegenskap så skulle man kunna hitta heltalsoptimum, då skulle vårt LP-optimum sammanfalla med vårt heltalsoptimum, men det är rimligt att anta att vårt problem inte har heltalsegenskap och således kommer vi förmodligen inte att hitta heltalsoptimum.

Det är lite knepigt att implementera branch and price, men det kan vara en god idé att göra i framtiden om man vill lösa heltalsproblemet till optimum.

2.2 Masterproblem

För att lösa masterproblemet ska följande matematiska modell lösas

- Låt $x_j = 1$ ifall ToD j används och 0 annars.
- $a_{ij} = 1$ ifall flyg i ingår i rutt j, 0 annars.
- c_j är kostnaden för att flyga ToD j.

$$\min z = \sum_j c_j \cdot x_j$$

då

$$\begin{aligned} \sum_j a_{ij} \cdot x_j &\geq 1 \qquad \forall i \\ x_j &\in \{0,1\} \qquad \forall j \end{aligned} \tag{2}$$

Till skillnad från modellen ovan ingår nu endast kolumner som har genererats i subproblemet.

2.3 Subproblem

Subproblemet går ut på att generera en ny ToD som kan användas för att lösa masterproblemet.

- Låt $use_{t,p} = 1$ ifall bågen mellan t och p ska användas, 0 annars.
- $arccost_{t,p}$ är kostnaden att lägga till bågen mellan t och p.

$$\text{min redCost} = \sum_{t,p} arccost_{t,p} \cdot use_{t,p}$$

då

$$\sum_{(s,node)} use_{s,node} = \sum_{(node,p)} use_{node,p} \qquad \forall node \tag{3}$$

$$\sum_{('START',p)} use_{'START',p} = 1 \tag{4}$$

$$\sum_{(s,'STOP')} use_{s,'STOP'} = 1 \tag{5}$$

$$use_{t,p} \in \{0,1\} \qquad \forall t,p$$

Modellen har tre intuitiva bivillkor. Ekvation (3) ser till att nodbalans skapas i varje nod, det vill säga att varje ToD blir en sammanhängande rutt. Ekvation (4) och (5) ser till att den ToD som genereras startar respektive slutar i Auckland.

3 Lösningsmetod

För att lösa vårt problem används följande lösningsmetod.

1. Börja med en artificiell kolumn. Det blir startlösningen till masterproblemet (Set Covering Problem, som benämns som ToD_Gen i den bifogade koden). En hög kostnad för denna startlösning väljs.
2. Lös masterproblemet (MP) som presenteras ovan.
3. Ta ut dualvariablen för bivillkoret i masterproblemet, det vill säga ekvation (2). Skapa nya kostnader på bågarna med den reducerade kostnaden, enligt:

$$arcCost_{t,p} = orgArcCost_{t,p} - \sum_{f} dualVar_{f}$$

4. Lös sub-problemet som är ett kortaste väg problem (Shortest Path Problem). Lösningen är en reducerad kostnad, på en ny kolumn.

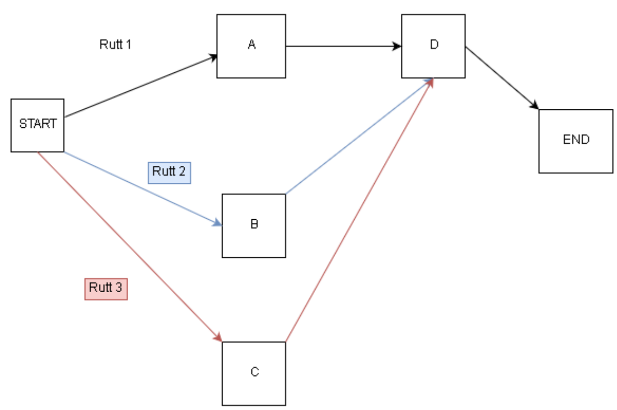
5. Om den reducerade kostnaden är < 0, lägg till den nya kolumnen i vår a-matris. Annars, om reducerad kostnad > 0, så stoppar vi och har då hittat LP-optimum för problemet.

6. Lös om problemet fast nu med heltalskrav.

Värt att poängtera är optimum av LP-relaxationen av problemet kommer att hittas, men det betyder inte nödvändigtvis att den heltalslösning som steg 6 ger är heltalsoptimum för problemet.

4 Paxning

Vi har räknat på två olika typer av paxningar. Den första räknar på hur många flygningar som det görs paxningar i och den andra räknar hur många besättningar som det görs paxningar för. Se nedan för ett kort exempel:



Figur 1: Exempel paxning

I rutt 2 och rutt 3 så kommer besättningen åka med flygplanet som kör i rutt 1, från nod D till nod END, alltså åker 2 st besättningar med från 2 olika rutter. Men det blir en flygning som det paxas i, nämnligen den från D till END.

5 Resultat

I uppgiften presenteras två olika problem. Dels ett litet problem med 35 flygningar och dels ett stort problem med 201 flygningar.

5.1 Lilla problemet

Modellen som presenteras ovan implementerades först för det lilla problemet. Den totala kostnaden för alla rutter blev 86 174 NZ. Totalt skapas det 38 stycken ToD och av dem används 9 i den slutgiltiga lösningen. Lösningen använder sig av paxningar i 6 st flygningar är 7 st besättningar som åker med i paxningarna. De har kostnadsfördelning enligt tabellen nedan.

Tabell 1: Kostnadsfördelning ToD

Rutt	Kostnad	Antal arbetspass	Antal flygningar
1	11699	6	6
2	10980	6	7
3	10043	4	5
4	7378	6	7
5	14116	6	6
6	8823	5	5
7	7722	4	4
8	14320	5	5
9	1090	3	3

Det är imponerande hur snabbt modellen löser det lilla problemet, det tar bara några få sekunder, ca. 7 sek, att nå optimum.

5.2 Stora problemet

Presentera resultat för stora problemet

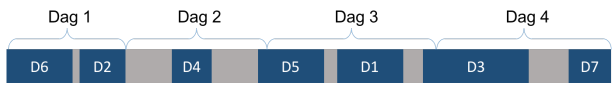
Det stora problemet tar naturligtvis längre tid att lösa, ungefär 1 min. Det får ändå anses vara en snabb lösning med tanke på att problemet är väsentligt större än det lilla problemet. Den totala kostnaden för alla rutter blev 406 828 NZ. Totalt skapas det 335 stycken ToD och av dem används 71 i den slutgiltiga lösningen. Lösningen använder sig av paxningar, för 32 st flygningar används paxningar och det blir 51 st besättningar som åker med i paxningarna.

6 Rostering problem

Nedan följer frågor kopplat till artikeln *Modeling and Solving the Crew Rostering Problem* av Caprara, Toth, Vigo och Fischetti.

6.1 Deluppgift a

I deluppgift a är uppgiften att skapa ett schema (roster) över de olika uppgifterna som ska utföras. Det spelar ingen roll vilken arbetsuppgift man startar med och det krävs ingen vila. Alla arbetsuppgifter kommer att ta fyra dagar att lösa. Se figur 2 nedan för att se ordningen. Det gråa fältet är då ingen uppgift görs, dvs väntetid.



Figur 2: Rostering utan krav på vila

6.2 Deluppgift b

I deluppgift b är uppgiften att skapa ett schema (roster) över de olika arbetsuppgifterna som ska utföras. Vi måste starta med Uppgift 4, men i övrigt finns det inga restriktioner kring ordning och till skillnad från deluppgift a krävs nu 18 timmars vila mellan arbetsuppgfiterna och 22 timmar om uppgiften ligger över natten. Figur 3 visar resultatet nedan visar resultatet.

Figur 3: Rostering med krav på vila

6.3 Deluppgift c

d_i^t används för att beteckna vila, där t är antal dagar som man vilar mellan i och efterföljande arbetsuppgift. Det vill säga,

$$d_i^t = t \cdot 1440 \tag{6}$$

För våra 7 arbetsuppgifter blir $d_i^t = t \cdot 1440 \; \forall i$, t kan väljas till 1,2 och 3, beroende på om man vill ha arbetspass som är direkt efterföljande varandra (för t = 1) eller arbetspass med ett vilopass emellan (t = 2) eller arbetspass med 2 vilopass emellan (t = 3).

c_{ij}^t definieras som den minsta tiden, i minuter, som är mellan start av arbetsuppgift i och start av arbetsuppgift j, när de är direkt efterföljande (i praktiken, i samma vecka), det vill säga

$$c_{ij}^t = (s_j + h \cdot 1440) - s_i \tag{7}$$

Där h är det minsta antal dagar som gör att man för en tillåten efterföljande sekvens på arbetsuppgifterna.

Våra olika c_{ij}^t får följande värden:

$C_{45}^1 = (s_5 + h \cdot 1440) - s_4 = 780$ min

$C_{62}^1 = (s_2 + h \cdot 1440) - s_6 = 660$ min

$C_{76}^1 = (s_6 + h \cdot 1440) - s_7 = 660$ min

$C_{71}^{1} = (s_1 + h \cdot 1440) - s_7 = 780$ min

$C_{45}^{2} = 780 + 1440 = 2200$ min

$C_{62}^{2} = 660 + 1440 = 2100$ min

$C_{76}^{2} = 660 + 1440 = 2100$ min

$C_{71}^{2} = 780 + 1440 = 2220$ min

$C_{45}^{3} = 780 + 2 \cdot 1440 = 3660$ min

$C_{62}^{3} = 660 + 2 \cdot 1440 = 3540$ min

$C_{76}^{3} = 660 + 2 \cdot 1440 = 3540$ min

$C_{71}^{3} = 780 + 2 \cdot 1440 = 3660$ min

7 Allmäna reflektioner

Det var initialt svårt att visualisera och förstå problemet. Vi kände att det hade kunnat vara bra att ha mer teoritisk bakgrund för att lösa problemet och förstå hur kolumngenerering fungerar. Annars var det ett roligt problem, det är kul med ett problem som har en tydlig anknytning till verkligheten. Desstuom kul med en artikel som man skulle läsa, trots att vi inte hann jobba med den så mycket som vi önskar. Kanske lite mycket “plug and play” över projektet, det vill säga att mycket av datan var färdigmanipulerad och för stort kodskellet. En idé skulle kunna vara att projektet börjar med en uppgift där man ska se till så att turerna som skapas (om de inte är givna) blir tillåtna med avseende på tid och andra faktorer.