Linköpings Tekniska Högskola
Matematiska institutionen/Optimeringslära
Kaj Holmberg

2020–09–11

# Projektinformation

## TAOP61 Optimering av realistiska sammansatta system

## Projekt 2: Optimering av elmotorutnyttjandet i en laddhybrid med dynamisk programmering

## 1 Uppgift

Uppgiften i projekt 2 är att optimera elmotorutnyttjandet i en laddhybrid. Metoden som ska användas är dynamisk programmering. Ni ska skriva ett eget program i Matlab/Octave eller Python med Numpy som gör beräkningarna.

## 2 Att göra

1. Studera beskrivningen av laddhybriden.
2. Studera informationen om Matlab/Octave eller Python.
3. Formulera problemet så att det passar att lösas med dynamisk programmering. Specificera alla nödvändiga definitioner (tillstånd, styrning, överföringsfunktion etc).
4. Implementera metoden i Matlab/Octave eller Python.
5. Lös problemen.
6. Redovisa genom att skriva en rapport om resultaten som kan läsas av både en som inte kan optimering och en som kan det och är nyfiken på hur ni gjorde. Inkludera definitionerna från punkt 3 ovan och en beskrivning av er metod och implementering.
7. Ev: Förbered en redovisning av resultaten/rapporten.

## 3 Problembeskrivning

En allmän beskrivning av en laddhybrid ges separat.

I detta projekt behandlas en något förenklad version av problemet. Vi har fyra inställningar för framdrivningssättet:
1. Bara el.
2. Mest el, med bidrag av bensin.
3. Lite el, med mera bensin.
4. Bara bensin.

Vägarna delas upp i fem kategorier:
1. Motorväg.
2. Landsväg.
3. Mindre väg, backig och/eller kurvig.
4. Väg i gles tätort (liten stad eller förort).
5. Väg i innerstad (trafikljus, många start och stopp).

Vi betecknar framdrivningssätt med index $k$, vägtyp med $t$, och vägavsnitt med $j$. Vi förutsätter att man har bestämt vilken väg man ska ta. Hela vägen delas upp i $n$ stycken vägavsnitt av olika typer och olika längder. Vägavsnitt $j$ har längden $l_j$ (m) och är av typen $t_j$.

Baserat på förväntad medelhastighet har man uppskattat hur mycket batterinivån sjunker om man färdas en meter på en väg av kategori $t$ med framdrivningssätt $k$, och det betecknas med $b_{kt}$. Dessutom har man beräknat en kostnad (inkluderande allt, dvs. bensin, slitage, miljöpåverkan etc.) för att färdas en meter på en väg av kategori $t$ med framdrivningssätt $k$, och den betecknas med $d_{kt}$.

Eftersom vägavsnitt $j$ har längden $l_j$, sjunker batterinivån med $a_{jk} = l_j b_{kt_j}$ om man kör det med inställning $k$, och kostnaden blir $c_{jk} = l_j d_{kt_j}$.

Vi använder variablerna $x_{jk} = 1$ om vi kör vägavsnitt $j$ med inställning $k$, och 0 om inte. Eftersom något av framdrivningssätten måste användas för varje vägavsnitt, får vi bivillkoren

$$\sum_k x_{jk} = 1 \text{ för alla } j$$

Om batteriet har laddning $L_0$ vid starten, får vi inte låta batterinivån sjunka med mer än det, så vi får följande bivillkor.

$$\sum_j \sum_k a_{jk} x_{jk} \leq L_0$$

Vi vill minimera kostnaden för hela resan, vilket ger målfunktionen

$$\min \sum_j \sum_k c_{jk} x_{jk}$$

Förutom detta återstår bara bivillkoren

$$x_{jk} \geq 0 \text{ för alla } j \text{ och } k$$

för att modellen ska bli komplett.

Denna modell kanske inte verkar så krånglig, men man bör beakta storleken. För att optimeringen ska ge ett bra resultat, bör uppdelningen i vägavsnitt vara ganska fin, dvs. många vägavsnitt behövs.

### 3.1 Indata

Data till detta problem består av koefficienterna $b_{kt}$ och $d_{kt}$. Matriserna $b$ och $d$ är ganska små, antal framdrivningssätt gånger antal vägtyper, i vårt förenklade problem 4x5. Dessa data återfinnes i filerna *linkbatt.txt* och *linkcost.txt*. Dessutom behövs längd $l_j$ och typ $t_j$ för varje vägavsnitt, vilket kan vara många. Dessa data återfinnes på filerna *path0.txt*, *path1.txt* etc. Dessa filer innehåller även startladdningen, $L_0$.

Man behöver läsa in dessa data och beräkna koefficienterna $a$ och $c$. (Detta sköts dock av tillgängligt program, se nedan.)

### 3.2 Instanser

Data har tagits fram för några olika instanser, dvs. vägsträckor.

| Instans | Antal sträckor | $L_0$ | Vad? |
|---|---|---|---|
| path1 | 3 | 10 | Litet test |
| path2 | 3 | 4000 | Mellantest |
| path3 | 5 | 4000 | Större test |
| path4 | 20 | 1000 | Från slumpad karta |
| path6 | 74 | 200000 | Linköping - Jönköping |
| path7 | 74 | 200000 | Linköping - Jönköping |
| path8 | 94 | 100000 | Östergötland, öst till väst |
| path9 | 246 | 100000 | Östergötland, norr till syd |
| path12 | 200 | 70000 | Slumpat |
| path13 | 300 | 100000 | Slumpat |
| path14 | 500 | 100000 | Slumpat |
| path15 | 700 | 200000 | Slumpat |
| path16 | 1000 | 300000 | Slumpat |

Man kan notera att antal sträckor avgör hur många steg man ska göra i dynamisk programmering, medan $L_0$ avgör hur stor tabellen blir i varje iteration. Skillnaden mellan de två första problemen är i princip bara storleken på $L_0$.

### 3.3 Kod

På filen *initeldynp.m* finns ett förslag på början och slut av matlab-koden. På rad 2 skriver man in namnet på datafilen med vägsträckorna, och på rad 3 skriver man en skalningsfaktor för diskretiseringen. Om den sätts till 1 beaktas varje heltaligt värde på batteriladdningen, och om den t.ex. sätts till 10 beaktas bara var tionde värde på batteriladdningen. Den ska helst vara lika med 1, men det går snabbare att lösa problem med stort $L_0$ (approximativt) om den ökas.

Därefter läses indatafilerna in. Detta behöver ni inte ändra på. Segment som får längd noll elimineras. Om batteriladdningen räcker till att köra på bara el hela vägen (dvs. om $L_0$ är för stort för att göra optimeringen intressant), sätts lösningen till detta (och problemet är löst).

Därefter beräknas lösningen för en dum förare som kör bara el (inställning 1) tills batteriet är slut, och därefter bara bensin (inställning 4). Kostnaden för denna lösning kan man jämföra sin bästa lösning med.

Efter detta ska ni skriva in er optimeringsalgoritm. Lösningen representeras av vektorn `xopt` som initieras till noll, men ska innehålla framdrivningssätt för varje segment. Dessutom ska `z` innehålla totala kostnaden och `s` batteriets laddning vid målgång.

Några tips (matlab-kommandon som kan vara bra att ha):

```
ix=ix-a;
```

Minskar indexvektorn `ix` med `a` enheter.

```
nix=1-ix>0;
```

`nix` blir 0 i de positioner där `ix` är större än noll och 1 i de positioner där `ix` mindre eller lika med noll.

```
ft0=circshift(ft1,[0 a]);
```

Vektorn `ft0` är vektorn `ft1` skiftad (flyttad) `a` steg åt höger (för radvektorer).

```
ft3(nix)=inf;
```

Elementen med index där `nix=1` i vektorn `ft3` sätts till ett stort tal. (De kommer därför aldrig att ge minimum.)

Efteråt summeras dels hur många segment som körs med varje inställning, och dels hur lång sträcka som körs med varje inställning. Detta behövar man inte ändra på.

# 4 Deluppgifter

1. Lös de små testproblemen *path1.txt* och *path2.txt* för att se att allt fungerar korrekt. Optimal kostnad för *path1.txt* är 97 med lösningen 3-4-4. Optimal kostnad *path2.txt* är 2144 med lösningen 1-1-3.

2. Lös de större problemen. Notera målfunktionsvärde och lösning, samt hur många segment och hur lång sträcka som körs med varje inställning. Jämför med kostnaden för en dum bilförare. Ange även skalfaktor (om den inte är ett).

3. Lös *path7.txt* (Linköping - Jönköping) med $L_0 = 250000$, 200000 (redan gjort), 150000, 100000, 50000 och 0, och studera hur kostnaden ökar, då den initiala batteriladdningen minskar. Notera även hur lösningstiden ändras.

4. I våra exempel innehåller filen *linkbatt.txt* inga negativa koefficienter, dvs. vi beaktar inte möjligheten att batterinivån *ökar*. I praktiken kan det dock vara så att ren bensindrift kan hjälpa till att ladda upp batteriet under vissa förhållanden. Hur skulle detta förändra modellen och metoden?

5. Ibland ändras förhållandena (väder, trafik) så att batterinivån sjunker mer eller mindre än förväntat. Om man under färden upptäcker att det är så, skulle man behöva lösa om optimeringsproblemet i bilen. Diskutera förutsättningarna för att använda er programvara på det sättet.