Optimering av elmotorutnyttjandet i en laddhybrid med dynamisk programmering

TAOP61 Projekt 2

HT2017 Linköpings Universitet

Adnan Avdagic, Carl-Martin Johansson, Joel Runesson

4 december 2017

Innehåll

Tabeller

1 Inledning

1.1 Bakgrund

Denna rapport är ett delmoment i kursen TAOP61 på Linköpings Tekniska Högskola. Uppgiften som rapporten handlar om berör elhybridbilar, vilkas elanvändning går att reglera för optimal framdrivning av fordonen beroende på vägtyp och trafikförhållanden.

För att göra problemet hanterligt och anpassa dess svårighet efter kursens mål, gjordes ett antal förenklingar och antaganden. Uppgiftens grundläggande antagande är att el är billigare att använda som drivmedel än bensin samt att elanvändning vid bilkörning antas leda till mindre miljöpåverkan än bensin. Vidare antas att ingen laddning får ske mellan start och mål. Sammantaget är det därför önskvärt att använda hela batteriets elladdning för den sträcka som ska köras, med andra ord ska batteriet vara tomt vid slutdestinationen. För enkelhets skull beaktas inte behovet av påfyllning av bensin, varför tillgänglig mängd bensin är obegränsad.

Körsträckan antas vara känd från början samt möjlig att indela i homogena delsträckor. Med homogen menas att hastigheten under hela delsträckan är konstant. Låga hastigheter och vägsträckor som kräver många start och stopp lämpar sig mer för eldrift. Detta har översatts till att hybridbilen har 4 möjliga framdrivningssätt att välja på, vilka presenteras i tabell 1.

Tabell 1: Olika framdrivningssätt

k	Framdrivningssätt
1	Bara el
2	Mycket el, lite bensin
3	Lite el, mycket bensin
4	Bara bensin

Slutligen antas att ett byte av framdrivningssätt enbart går att göra mellan två delsträckor. Alla delsträckor kan delas in i olika kategorier (betecknas t) baserat på vägens beskaffenhet. Detta framgår av tabell 2:

Tabell 2: Kategorier av delsträckor

t	Kategori
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

2 Syfte

Syftet med projektet är att formulera en optimeringsmodell som optimerar hur bilens framdrivningssätt ska se ut givet olika förutsättningar för att minimera kostnaden och få ut maximal nytta av den laddade elen i batteriet vid start. De förutsättningar som ändras är bland annat ändrade körsträckor och andel tillgängliga el-enheter i batteriet. Alla givna varianter på problemet, vilka är 16 stycken, optimeras för att räkna ut vilket framdrivningssätt bilen ska ha under alla olika sträckor i de specifika fallen, vilket redovisas senare under resultatdelen. Vårt intresse är att se hur detta kappsäcksproblem kan lösas med dynamisk programmering.

3 Metod

För detta optimeringsproblem var lösningsmetoden dynamisk programmering given av uppgiftsbeskrivningen. Dynamisk programmering är en metod för att lösa optimeringsproblem via rekursion. Delproblem löses genom att tablåer skapas där man för varje tidssteg skapar en ny tablå. Det finns det ett antal varianter av dynamisk programmering att välja bland: vägproblem, kappsäcksproblem samt lagerhållningsproblem. Strukturen på detta problem kännetecknas av möjligheten att skriva problemet med endast ett bivillkor, varför varianten kappsäcksproblem valdes eftersom definitionen av ett sådant är just att det bara finns ett bivillkor. Bivillkoret i denna uppgift är att summan av alla delsträckors batterisänkningar inte får överskrida den mängd batterienheter som finns tillgängligt.

För kappsäcksproblem är tanken att se hur den kan fyllas på bästa sätt, genom att föregående tablås bästa värden jämförs med nuvarande värden. När alla tablåer arbetats igenom används rekursion för att se vilken väg som använts för varje steg och då fås det optimala svaret.

3.1 Definitioner

Styrning: Ordet styrning används i dynamisk programmering för att beskriva hur ett visst tillstånd nåddes, det vill säga från vilken nod man kom ifrån. För detta optimeringsproblem betyder styrning vilket framdrivningssätt som användes på delsträckan j.

"Stupid driver": I det fördefinierade läget "stupid driver" körs bilen endast med el vid start tills alla laddningsenheter är slut. Därefter används uppenbarligen bara bensin. Detta läge används senare i rapporten då vi jämför det med våra framräknade optimala lösningar.

Tillstånd: I dynamisk programmering används s_k som notation för tillstånd. Tillstånd brukar ses som lagernivån efter tidsperiod k. I denna uppgift utgörs tillstånd av den mängd batteri som finns att tillgå för varje delsträcka där tidsperioden k är delsträckan j i den här uppgiften.

Överföringsfunktion: Dynamisk programmering utför iterativa beräkningar för att erhålla optimum. Dessa beräkningar använder sig av en överföringsfunktion som kopplar ihop ett tidssteg med nästa, vilket i detta fall blir kopplingen mellan två delsträckor. Matematiskt ser därför överföringsfunktionen ut så här:

$$s_{k-1} = s_k - a_{jk} x_{j,k} \quad \forall\, j$$

Målfunktionsvärde: är det lägsta värdet av de möjliga värdena för varje tillstånd s_k.

$f_k(s_k) = min\{c_{jk} + f_{k-1}(s_{k-1}(x_{jk}))\}$ (givet att man kan använda s_k antal batterienheter) (3.1)

Randvillkor: Eftersom en tablå i dynamisk programmering beräknas med hjälp av målfunktionsvärdet (3.1) från föregående tablå, måste ett villkor sättas för tablå 0:s målfunktionsvärde. Detta sätts till:

$$f_0(s_0) = 0 \quad , \quad s_0 \geq 0 \quad , \quad s_N = b$$

4 Resultat

Nedan listas tabell 3 som visar de olika instansernas körda delsträckor med olika drivsätt, hur långt dessa drivsätt totalt kört, den optimala kostnaden samt kostnaden för "Stupid driver":. Tabell 4 visar sedan specialfall för Path 7 för att se hur målfunktionsvärdet och lösningstiden ändras beroende på initial laddning.

Tabell 3: Resultat

			Antal delsträckor				Total längd			
Instans	Stupid Driver [sek]	Målfunktionsvärde [sek]	k=1	k=2	k=3	k=4	k=1	k=2	k=3	k=4
Path 1	112	97	0	0	1	2	0	0	2,5	2,5
Path 2	3 506	2 144	2	0	1	0	328	0	227	0
Path 3	1 598	1 269	1	3	1	0	53	237	169	0
Path 4	1 086	690	10	4	1	5	76,8	35,7	3,1	40,8
Path 5	2 151	1 507	3	4	8	8	18,5	74,5	64,9	46,3
Path 6	123 499	74 235	33	9	7	25	13 424	3 376	923	6 532
Path 7	113 857	79 581	35	10	10	19	9 241	2 936	1 016	11 062
Path 8	221 676	175 174	11	4	35	44	2 052.2	566.6	8548	14638.7
Path 9	172 676	128 986	109	19	28	90	5 933.7	1 863.9	5 746.6	7 605.9
Path 12	79 738	55 679	100	9	11	80	5 954	209.1	721.3	4 762.9
Path 13	127 164	91 023	138	19	13	130	8 334.2	1 000.2	893.4	7 527.3
Path 14	292 141	230 521	49	82	150	219	3 053.4	5 155.1	9 053.7	12 750.6
Path 15	343 506	252 991	234	46	106	314	14 391.6	2 701.9	6 465.5	18 516.4
Path 16	488 578	359 353	368	61	119	452	22 858.2	3 914.3	7 704	26 709.4

Tabell 4: Path 7 - Olika scenarion

Batteri vid start [batterienheter]	Målfunktionsvärde [sek]	Lösningstid [s]
250 000	47 237	83.407413
200 000	79 581	64.595474
150 000	117 081	48.247987
100 000	156 147	32.409360
50 000	203 268	16.746198
0	287 388	0.024475

5 Diskussion

5.1 Optimal kostnad jämfört med kostnaden för en "stupid driver"

Målfunktionsvärdet, som är det optimerade värdet från modellen, kan noteras vara lägre än kostnaden för "stupid driver" för samtliga instanser av problemet, vilket kan ses i tabell 3. Detta är rimligt då det optimala värdet aldrig kan bli högre än kostnaden för "stupid driver" utan kostnaden kan som sämst bli lika stor.

5.2 Kostnadsförändring vid ändrad initial batteriladdning

Ju mindre batteri som finns tillgängligt vid start, desto högre kommer målfunktionsvärdet bli. Detta kan ses i tabell 4 där målfunktionsvärdet är cirka 47 000, 80 000, 117 000, 156 000, 203 000 och 287 000 då batteriet startar med 250 000, 200 000, 150 000, 100 000, 50 000 respektive 0 batterienheter tillgängliga. Det vill säga, kostnaden ökar med över 500 % om bilen inte startar med något laddat batteri, exempelvis om föraren glömt att ladda batteriet innan start, jämfört med om bilen får starta med 250 000 batterienheter. Intressant är att modellens optimeringstid däremot starkt minskar då tillgängligt batteri vid start minskar. Exempelvis tar det cirka 0,02 sekunder att lösa problemet om batteriet startar på noll jämfört med cirka 83,41 sekunder att lösa problemet om batteriet startar med 250 000 batterienheter. Detta beror på att tiden och antalet tillstånd per steg är proportionella mot varandra.

5.3 Modellförändringar för uppladdning vid bensindrift

Som tidigare beskrivits är modellen en förenkling av verkligheten. Hybridbilar har dock möjlighet att ladda batteriet då de kör på enbart bensin. Detta är inget vår modell klarar av, utan för att att implementera detta måste denna modell tyvärr ändras.

Om uppladdning av batteriet tillåts vid ren bensindrift innebär det förändringar i matrisen a_{jk} som innehåller data över batteriminskningen vid olika framdrivningssätt. Förut var kolumnen som reglerar körning med enbart bensin fylld av nollor, då ingen batteriminskning sker när bara bensin används. Om batteriet däremot kan laddas upp kommer kolumnens element i a-matrisen bli negativa, vilket leder till att framtida tablåer ej kan räknas ut. Det beror på att vid uträkning av målfunktionsvärdet $f_k(s_k)$ används föregående tillstånd s_{k-1} som i sin tur räknas ut med hjälp av a-matrisen. Negativa element i a-matrisens högraste kolumn leder till att vid skapandet av $f_k(s_k)$ i tablån, flyttas $f_k(s_k)$ åt vänster istället för åt höger (som i fallet vid positiva element i a-matrisen). Modellen försöker därmed räkna ut värden på platser i tablån som inte existerar.

Även kostnadsmatrisen c_{jk} skulle behöva justeras för att ta hänsyn till att mer bensin åtgår då batteriet laddas vid ren bensindrift.

5.4 Modellförändringar vid oväntade förhållanden

För att göra problemet ännu mer verklighetstroget kan hänsyn tas till att yttre faktorer såsom motvind och trafik kan påverka batterinivån så att den ursprungliga beräkningen av optimalt framdrivningssätt mellan start och mål är inaktuell. I sådana fall behöver en ny optimering genomföras. Detta kan vår modell redan ta hänsyn till. Anta till exempel att en ny optimering behöver genomföras innan vägsträcka j = 11. Detta hade kunnat vara problematiskt för modellen om det inte vore för att målfunktionsvärdet $f_k(s_k)$ nollställs inför varje ny optimering av det fullständiga problemet.

En viktig faktor som inte får glömmas bort är att optimeringsproblemet ska lösas i bilen vilket förutsätter att lösningstiden är relativt snabb. För de största instanserna löstes problemet på ca 20 minuter. För att snabba upp tiden kan en så kallad skalfaktor justeras. En skalfaktor $\neq 1$ innebär att hänsyn inte tas till samtliga heltalsvärden på batterinivån L_0. Eftersom batterinivån är den faktor med störst inverkan på lösningstiden leder detta till att tiden minskar då färre värden på batteriladdningen beaktas. Dock ökar osäkerheterna i modellen med ökad skalfaktor vilket gör att optimum inte nås.