Linköpings Tekniska Högskola
Matematiska institutionen/Optimeringslära
Kaj Holmberg

2021–09–03

Laborationsinformation

TAOP88 Optimering för ingenjörer, HT21

1 Laboration 6: Implementering av simplexalgoritmen

Uppgiften i laboration 6 är att implementera (en enkel variant av) simplexalgoritmen, samt att med den lösa några LP-problem av olika storlek.

1.1 Specifikation inför implementeringen

Vi föreslår att Python eller MATLAB/Octave används, då ett annat val av språk (Java, ADA, C, C++, Fortran) förmodligen skulle göra uppgiften väldigt jobbig. I laborationsinformationen ingår en liten lathund för Python och MATLAB/Octave och ni bör kunna klara er med det som finns där. Poängen är att dessa språk enkelt klarar av vektor- och matrisoperationer.

Programmet ska kunna läsa en datafil med godtyckligt namn (`<file>.mat`) som innehåller data för problemet. Inladdning av datafilen ger som resultat matriserna/vektorerna A, b och c som motsvarar beteckningarna i standardmodellen för ett LP-problem,

$$\min c^T x \quad \text{då } Ax = b, x \geq 0.$$

Vidare laddas en vektor *bix* in, innehållande kolumnindex för basvariablerna för en tillåten startbas. Ni kan alltså utgå ifrån att ni har en tillåten startbas att börja med. Det är inte säkert att *bix* innehåller slackvariablerna, så starttablån kan behövas fixas till. (Vektorerna b och c läses in som kolumnvektorer och *bix* som en radvektor.) Slutligen fås också (oftast) *xcheat* (kolumnvektor) och *zcheat* (skalär) som innehåller optimallösning och optimalt målfunktionsvärde. Alla testproblem är minimeringsproblem.

Man ska för varje iteration kunna se (index för) inkommande och utgående variabel samt målfunktionsvärdet. Obegränsad lösning ska kunna upptäckas. Funnen optimalpunkt ska redovisas med lösning x och målfunktionsvärde z. Skriv ut skillnaden mellan z och *zcheat* samt normen av skillnaden mellan x och *xcheat*. Total exekveringstid (exkl I/O) ska mätas. Tips: Vid kontroll av optimalitetskriterierna, jämför inte reducerade kostnader med exakt 0, utan tillåt en tolerans på 10^{-6} för att undvika numeriska problem.

Förberedelse: Tänk igenom och skriv ner programmets struktur och de olika operationer som måste göras (såsom val av inkommande resp utgående variabel, basbyte mm). Använd den tillgängliga informationen (lathund för MATLAB, tips mm). Gör ett enkelt blockschema.

1.2 Laborationsuppgifter

Följande LP-problem skall lösas med den egna implementeringen, och total exekveringstid (exkl I/O) ska mätas och redovisas för varje problem. Alla problem är minimeringsproblem. Antalet variabler som anges nedan inkluderar slackvariablerna.

Problem	n	m	Kommentar	Målfunktionsvärde	Tid
`lab1-cloetta.mat`	5	3	Från boken, kap 2.1		
`lab1-mus.mat`	5	3	Från föreläsning		
`lab1-p39.mat`	7	5			
`lab1-obegr.mat`	4	2	Obegränsad lösning		
`lab1-small.mat`	35	23			
`lab1-israel.mat`	316	174	Numeriskt krävande		
`lab1-big.mat`	1400	400			

Problemdata kan hämtas från hemsidan http://courses.mai.liu.se/GU/optlab-information/. (Håll reda på var datafilerna hamnar. Det är enklast om MATLAB startas från denna katalog.)

Ett hedersomnämnande utdelas till konstruktörerna av den implementering som snabbast löser problemet `lab1-big.mat`. (Problemet `lab1-israel.mat` ger ibland numeriska svårigheter. Det är inte ett absolut krav att er kod löser detta problem korrekt.)

1.3 Redovisning

Följande delar ska redovisas för laboration 6.
- Optimala målfunktionsvärden samt exekveringstider för problemen i avsnitt 1.2.
- En principiell beskrivning av er implementering.
- Kommenterad Python/MATLAB/Octave-kod.

Avsluta laborationen med att skicka in er Python/MATLAB/Octave-kod i Lisam, tillsammans med information om hur man kör programmet. Försök att ge programmet/filen ett unikt namn.

1.4 Några tips

En möjligtvis knepig detalj i implementeringen är indexhanteringen, dels vid val av utgående variabel och dels vid basbyte. Förutom basindexvektorn *bix*, kan det vara bra att jobba med en vektor med index för ickebasvariablerna, *nix*. Observera att *bix* även anger kopplingen mellan raderna i basmatrisen och basvariablerna, och att *nix* anger vilken variabel som står på en viss position i vektorn med ickebasvariabler.

I varje iteration måste man hålla reda på index för inkommande variabel, t ex *inix*, och utgående variabel, t ex *utix*. Vid basbyte kan man t ex göra `bix[utix]=nix[inix];` och sedan uppdatera *nix*. (Det spelar ingen roll i vilken ordning ickebasvariablerna kommer, bara man vet vilken ordning det är.) Därefter uppdateras B, N, c_B och c_N enligt kända formler.

1.5 Simplexmetoden i principiell form

1. Skaffa en tillåten startbas. Partitionera x i basvariabler och ickebasvariabler, vilket ger partitionering av A och c.
2. Beräkna eller uppdatera basinversen.
3. Beräkna högerledet (variablernas värden), målfunktionsvärde och skuggpriser (dualvariabler).
4. Beräkna reducerade kostnaderna och avgör om optimum är funnet, eller välj inkommande variabel.
5. Beräkna inkommande kolumn, anpassad till aktuell bas, och avgör om problemet har obegränsad lösning, eller välj utgående variabel.
6. Byt bas, dvs uppdatera partitioneringen av x samt A och c. Gå till 2.