Linköpings Tekniska Högskola
Matematiska institutionen/Optimeringslära
Kaj Holmberg

2022–09–02

Laborationsinformation

1 Laboration 4-5: Lösning av heltalsproblem med hjälp av modelleringsspråk och heuristik

Laboration 4 syftar till att ge övning i att lösa strukturerade heltalsoptimeringsproblem med hjälp av ett generellt modelleringsspråk, samt ge viss insikt i relationerna mellan heltalsproblem och deras LP-relaxation. Dessutom skall laborationen påvisa möjligheterna att utnyttja ett problems specifika struktur, dels genom att modifiera problemformuleringen så att en bättre LP-relaxation erhålls, och dels, i laboration 5, genom att utveckla en alldeles egen heuristik för problemet.

Modelleringsspråket vi använder i laboration 4 heter GMPL, tillsammans med en optimeringslösare, glpsol. En kort beskrivning inkluderande illustrativa exempel ges separat. Heuristiken i laboration 5 implementeras troligtvis enklast i MATLAB eller det likvärdiga Octave.

Med hjälp av GMPL kan man beskriva ett optimeringsproblem med generella beteckningar. Instanser av problemet med olika storlekar och olika data kan sedan importeras och lösas. Paketet GLPK, som innehåller GMPL och glpsol, är fritt nedladdningsbart. (Detta behövs inte om ni gör labben på LiU.)

Observera att laboration 4 görs genom att skapa en modellfil (med valfri texteditor) och ge kommandon i ett terminalfönster, dvs. på det gamla hederliga sättet att använda dator.

1.1 Problemställning

Det optimeringsproblem som ska behandlas i denna laboration är det kapaciterade lokaliseringsproblemet. Låt m vara antalet möjliga platser för lokalisering av anläggningar (t.ex. fabriker), n antalet kunder, d_j efterfrågan hos kund j, s_i kapaciteten hos en anläggning på plats i, f_i den fasta kostnaden för att öppna en anläggning på plats i, c_{ij} transportkostnaden per enhet till kund j från anläggningsplats i, och e en diskonteringsfaktor. Samtliga koefficienter kan antas vara positiva.

Variabeldefinition: $y_i = 1$ om en anläggning placeras på plats i, och 0 om inte.
x_{ij} = transporterad mängd från anläggning på plats i till kund j.

Problemet att finna lösningen med lägst totalkostnad kan modelleras på följande sätt.

$$
\begin{aligned}
v^* = \quad \min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} ef_i y_i & & & \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} \;\leq\; s_i y_i & i = 1,\ldots,m & (1) \\
& \sum_{i=1}^{m} x_{ij} \;=\; d_j & j = 1,\ldots,n & (2) \\
& x_{ij} \;\geq\; 0 & i = 1,\ldots,m, j = 1,\ldots,n & (3) \\
& y_i \;\in\; \{0,1\} & i = 1,\ldots,m & (4)
\end{aligned}
\qquad \textbf{(P1)}
$$

Ofta avser egentligen kostnaderna olika tidsperioder. De fasta anläggningskostnaderna kan avse nybyggnation idag, medan transportkostnaderna kan avse samlade transportkostnader över en längre period (kanske flera år). Man måste då göra dessa kostnader jämförbara, och diskonteringsfaktorn e kan användas för detta. Man kan även prova olika scenarior (för t.ex. olika nivå på räntan). De första körningarna ska göras med $e = 1$.

Ibland finns en konflikt mellan fasta och rörliga kostnader. En plats med låga transportkostnader ligger troligen centralt, vilket leder till höga priser för mark och byggnation, dvs. höga fasta kostnader. Låga fasta kostnader uppträder troligen långt från kunderna, vilket medför höga trasnportkostnader.

En annan svårighet är att en anläggning antingen öppnas i sin helhet eller inte alls, även om bara en liten del av dess kapacitet behöver utnyttjas. LP-relaxationen av problemet är lättare att lösa, men ger ofta orealistiska lösningar där delar av anläggningar öppnas.

1.2 Förberedelser

Resultat och svar på frågor skrivs ner på separat resultatblad. Redovisning av lab 4 sker med Lisam Test. Gör gärna detta medan laborationen görs. Observera möjligheten att kontrollera svaret direkt.

1. Betrakta den matematiska formuleringen P1. Hur många variabler och bivillkor finns i denna formulering, om vi har m platser för fabriker och n kunder? Specificera för floc6 och floc8. (Man räknar inte med $x \geq 0$ eller $x \in \{0,1\}$.)

2. Lab 4: Utgå ifrån formuleringen ovan och förbered en GMPL-modell. Nedan ges parameterdefinitioner att infoga i er GMPL-modell:

   ```
   param n;
   param m;
   param s{1..m};
   param d{1..n};
   param f{1..m};
   param c{1..m, 1..n};
   param e := 1;
   ```

   Komplettera modellen med övriga definitioner, dvs. variabeldefinition, målfunktion och bivillkor.

3. Lab 4: Fundera över uppgift 3 i avsnitt 1.3. Hur många bivillkor får modellen P2? Jämför med svaret i uppgift 1 ovan. Specificera för floc6 och floc8.

4. Lab 5: Hitta på och beskriv en egen girighetsheuristik för detta problem. Man ska försöka minimera den totala kostnaden, så det är inte tillåtet att strunta i anläggningskostnaderna eller transportkostnaderna. Tänk igenom implementeringen. Ett skal för koden finns på sidan
`http://courses.mai.liu.se/GU/optlab-information/`.

5. Lab 4 och 5: Fundera på svaret till följande fråga: Kommer värdet på diskonteringsfaktorn e att påverka hur bra heuristiken är jämfört med den optimala lösningen?

1.3 Laborationsuppgifter

Ni ska lösa följande instanser av lokaliseringsproblemet.

Name	m	n
floc1	3	5
floc2	12	18
floc3	10	25
floc6	20	100
floc7	30	150
floc8	30	200

Dessa är givna i GMPL-datafiler med namnen floc1.dat, floc2.dat etc. Det finns även (med tanke på laboration 5) MATLAB-filer, floc1.m, floc2.m, osv, där indata tilldelas på ett sätt som kan användas i MATLAB, samt datafiler i MATLABs interna format, floc1.mat, etc, som kan läsas in med kommandot load i MATLAB.

Redovisning av laborationerna består helt enkelt av att redovisa förberedelserna samt resultaten av punkterna nedan. För lab 4 sker detta i Lisam Test. För lab 5 skrivs en kort rapport. Alla skall dessutom skicka in sin heuristik (MATLAB/Octave/python-kod) med email till Kaj, kaj.holmberg@liu.se och till labassistenten. Försök att ge filen ett unikt namn, och ange hur man kör programmet.

Problemdata 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.)

Laboration 4:

1. Använd GMPL och glpsol för att lösa de olika instanserna av P1. Ange optimala målfunktionsvärden (dvs. minimala totalkostnader). Fyll i tabellerna på resultatbladet. Hur många trädsökningsnoder ("noder" i tabellen, se till höger i utskrifterna från glpsol) har genererats för varje instans? Hur många simplexiterationer ("iter" i tabellen, se till vänster i utskrifterna från glpsol) har gjorts? Hur lång tid tog lösandet?

2. Lös *LP-relaxationen*, LP1, till dessa instanser genom att ta bort heltalskravet i modell P1. (Behåll övre gränsen 1 på y-variablerna.) LP-relaxationen ger en undre

gräns till v^*. Hur stor är skillnaden i totalkostnad mellan heltalsoptimum och dess LP-relaxation? Den viktigaste delen av en lösning är vilka anläggningar som skall öppnas. Vilken information om detta ger LP-relaxationen? Studera och skriv upp y-lösningarna.

3. För ett heltalsproblem är det önskvärt att LP-relaxationen ger en uppskattning som ligger så nära heltalsoptimum som möjligt. Utgå från modell P1 och lägg till följande grupp av bivillkor. Kalla den nya modellen P2.

   $$x_{ij} \leq d_j y_i \quad \forall i, j$$

   Dessa bivillkor kan ge en starkare LP-relaxation (dvs. värden på y som ligger närmare 1 när de inte kan vara 0).

4. Lös LP-relaxationerna, LP2, med modell P2. Ger denna LP-relaxationen bättre uppskattningar och information? Jämför y-lösningarna med de i uppgift 2.

5. Lös heltalsproblemet med modell P2. Notera antalet trädsökningsnoder och antalet simplexiterationer, och jämför med resultatet i uppgift 1.

6. Lös floc3 med P1 för $e = 0.01$, 0.1, 1 (redan gjort), 10 och 100. Notera målfunktionsvärden, samt hur många anläggningar som öppnas i optimallösningen.

Laboration 5:

Applicera er heuristik på dessa instanser genom att skriva ett litet datorprogram (MATLAB/Octave/Python rekommenderas). Skriv in kursens namn och era namn som en kommentar först i filen. Beskrivningen av metoden kan göras som utförliga kommentarer i koden eller separat. Hur bra lösningar hittar er heuristik jämfört med heltalsoptimum? Notera tiden programmet tar.

Man kan göra ett generellt program som förutsätter att data redan finns, och sedan börja med att anropa t.ex. floc1.m. Det går dock snabbare (för stora problem) att läsa en mat-fil.

Lös sedan floc3 med heuristiken för $e = 0.01$, 0.1, 1 (redan gjort), 10 och 100. Notera målfunktionsvärden, samt hur många anläggningar som öppnas. Jämför resultatet med de optimala lösningarna.

För vilka värden på e fungerar heuristiken bäst?

Ge en kort beskrivning av er heuristik. Gör även en bedömning av den. Vad är bra? Vad är dåligt? Vad kan förbättras? Vilka problemegenskaper passar den bättre/sämre för? (Stora/små fasta kostnader, stora/små kapaciteter, etc.) Skriv detta på baksidan av resultatbladet.