

# Korteste veje med betingelser

## Dat2A godkendelsesopgave 2

Martin Zachariasen

13. april 2004

### 1 Formalia

Dette er den anden af to godkendelsesopgaver på kurset Dat2A, efterår 2003 og forår 2004. Opgaven skal løses i grupper på 2–3 personer (enkeltmandsgrupper kræver begrundelse og skal godkendes af en instruktør).

Opgaven bliver stillet onsdag d. 14/4 2004 kl. 11 og skal afleveres i førstedelsadministrationen i 2 eksemplarer inden **fredag d. 30/4 2004 kl. 11**. Bedømmelserne foreligger i førstedelsadministrationen senest fredag d. 14/5 2004.

Opgaven vil blive bedømt enten *godkendt*, *ikke godkendt* eller *dumpet*. Bedømmelsen *ikke godkendt* giver mulighed for genaflevering senest d. 21/5 2004, mens bedømmelsen *dumpet* ikke giver mulighed for genaflevering. Opgavebesvarelser hvori der ikke er gjort et *væsentligt forsøg* på at besvare alle spørgsmål bliver betragtet som dumpet.

Rapporten skal være på højst 8 sider, excl. programudskrifter og udskrifter fra kørsler (der også vedlægges). Desuden skal alle kildetekstfiler — herunder en README-fil der beskriver hvordan filerne oversættes til et færdigt program — gøres tilgængelige på en 1. delskonto (di-konto) ejet af en af deltagerne; filerne lægges i et katalog med navnet G22A og må ikke rettes efter rapportens aflevering.

## 2 Formål

Formålet med opgaven er at

- få et indblik i teknikker til løsning af NP-hårde problemer
- forstå betydningen af kvaliteten af grænseværdier i branch-and-bound algoritmer
- få praktisk erfaring med implementation af grafalgoritmer

Opgavens formål er at implementere en branch-and-bound algoritme til løsning af det betingede korteste vej problem (se afsnit 3). Desuden skal et antal relaterede teoretiske spørgsmål besvares.

Branch-and-bound algoritmen implementeres i C++ og gerne ved anvendelse af LEDA. Besvarelsen skal bestå af et tekstafsnit, hvori nedenstående opgaver besvares, samt udskrift af det **kommenterede** program. Ved bedømmelse af opgaven vil der blive lagt vægt på at alle spørgsmål (hver for sig) er besvaret tilfredsstillende.

## 3 Det betingede korteste vej problem

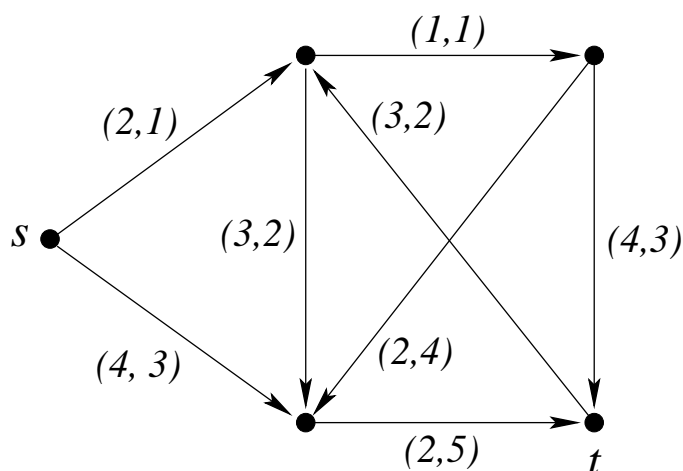
Vi betragter en orienteret graf  $G = (V, E)$ . Til hver kant  $(u, v) \in E$  er der knyttet en vægt  $w(u, v) \geq 0$  og en forsinkelse  $a(u, v) \geq 0$  (begge reelle tal). Lad  $p = \langle s, \dots, t \rangle$  være en vej i  $G$  fra en given knude  $s \in V$  til en given knude  $t \in V$ , og lad  $w(p)$  betegne den samlede vægt på  $p$  og lad  $a(p)$  betegne den samlede forsinkelse på  $p$ .

Det *betingede* korteste vej problem er nu at finde en vej  $p$  fra  $s$  til  $t$  således at vægten  $w(p)$  er mindst mulig samtidig med at  $a(p) \leq A$ , hvor  $A$  er en given maksimal forsinkelse. Hvis vi lader  $P(s, t)$  betegne mængden af alle simple veje fra  $s$  til  $t$ , kan problemet opskrives således:

$$\min_{p \in P(s, t), a(p) \leq A} w(p)$$

Problemet er eksempelvis relevant i forbindelse med kommunikation i netværk, hvor  $w(p)$  kan angive omkostningen ved at benytte en given kommunikationsforbindelse  $p$ , og hvor  $a(p)$  er signalforsinkelsen over forbindelsen — som ikke må være større end  $A$ .

Figuren herunder viser et eksempel, hvor der for hver kant er angivet vægt og forsinkelse  $(w, a)$ . Den maksimale forsinkelse er  $A = 5$ . Bemærk at i dette tilfælde er den korteste vej fra  $s$  til  $t$  (som har samlet vægt 6) ikke lovlig, idet dens forsinkelse overskrider den maksimale på 5.



## 4 NP-fuldstændighed

Det sædvanlige (ubetingede) korteste vej problem kan som bekendt løses i polynomiel tid med Dijkstras algoritme. Det samme gælder ikke for det betingede problem.

**Opgave 1** Formuler afgørlighedsversionen af det betingede korteste vej problem. Giv et detaljeret bevis for at afgørlighedsversionen er NP-fuldstændig. Benyt en reduktion fra subset-sum problemet (se afsnit 34.5.5 i CLRS [1]). ■

## 5 Nedre grænseværdi

Vi ønsker at løse det betingede korteste vej problem ved anvendelse af branch-and-bound. Her gives en nedre grænseværdi baseret på Lagrange relaxering: Lad  $\lambda \geq 0$  være et vilkårligt ikke-negativt tal. Definer  $\mathcal{L}(\lambda)$  som følger:

$$\mathcal{L}(\lambda) = \min_{p \in P(s,t)} (w(p) + \lambda(a(p) - A))$$

**Opgave 2** Bevis at  $\mathcal{L}(\lambda)$  er en nedre grænseværdi for *enhver* værdi af  $\lambda \geq 0$ . Vis at  $\mathcal{L}(\lambda)$  kan findes ved at løse et (ubetinget) korteste vej problem. ■

**Opgave 3** Implementer en algoritme til udregning af denne grænseværdi, og find herunder en fornuftig metode til iterativt at forbedre (forøge) grænseværdien ved at ændre  $\lambda$ -værdien. ■

## 6 Branch-and-bound algoritme

Branch-and-bound algoritmen skal benytte dybde-først søgning, der kan implementeres som en rekursiv procedure som skitseret nedenfor (hvor  $z^*$  angiver den bedste lovlige løsning/øvre grænse):

```
CSP_BRANCH( $G, s, t, w, a, A$ )
  Find en nedre grænseværdi  $\ell$  for problemet.
  if  $\ell \geq z^*$  then return
  if løsnings fra grænseværdiberegningen netop har forsinkelsen  $A$  then
    Gem løsningen, opdater  $z^*$ .
  else
    Vælg en udgående kant  $e = (s, v)$  fra  $s$ .
    Delproblem 1:  $e$  skal være med i løsningen:
      CSP_BRANCH( $G \setminus \{s\}, v, t, w, a, A - a(s, v)$ )
    Delproblem 2:  $e$  skal ikke være med i løsningen:
      CSP_BRANCH( $G \setminus \{e\}, s, t, w, a, A$ )
```

Håndteringen af “levende knuder” i branch-and-bound træet foretages automatisk af rekursionen, idet tidligere problemer gemmes på stakken. Bemærk at i det første delproblem bliver kanten  $e = (s, v)$  medtaget i løsningen ved at slette  $s$  fra  $G$  og derefter finde en korteste betingede vej fra  $v$  til  $t$ , hvor den maksimale forsinkelse er  $A - a(s, v)$ .

**Opgave 4** Implementer branch-and-bound algoritmen som skitseret ovenfor. Beskriv fordele og ulemper ved forskellige strategier for valg af kanten  $(s, v)$ , og implementer den strategi, som I mener vil fungere bedst. Ved udregning af den nedre grænseværdi v.h.j.a. Lagrange relaxering skal følgende mulige strategier for fastsættelse af  $\lambda$  undersøges:

- Der benyttes det samme  $\lambda$  i alle knuder i branch-and-bound træet (f.eks. det der findes i rodknuden).
- Der benyttes forskellige værdier for  $\lambda$  i de enkelte knuder.
- Der benyttes  $\lambda = 0$  i alle knuder.

Den sidste strategi betyder at forsinkelsesbegrænsningen droppes i forbindelse med grænseværdiberegningen. ■

## 7 Eksperimenter

Afprøv den konstruerede algoritme på de probleminstanser, som er tilgængelige i kataloget `~dat2a/G2`. Filen `~dat2a/G2/read_csp.cc` indeholder en indlæsningsprocedure samt tilhørende testprogram, og filen `~dat2a/G2/README` indeholder en oversigt over alle probleminstanser samt deres optimale løsninger.

**Opgave 5** Rapporter: øvre og nedre grænseværdier i rodknuden, antal knuder i branch-and-bound træet, samt total beregningstid. (Hvis det tager mere end 5 minutter at løse en probleminstans, så rapporter blot antal knuder og de bedste nedre og øvre grænser for den optimale løsning efter 5 minutter.) Kommenter resultaterne. ■

## 8 Hurtigere beregning af nedre grænseværdi

I forbindelse med beregning af en god nedre grænseværdi skal  $\mathcal{L}(\lambda)$  findes for mange forskellige værdier af  $\lambda$ .

**Opgave 6** Overvej om man i praksis kan undgå at foretage en komplet (ubetinget) korteste vej beregning hver gang man ændrer  $\lambda$ . Diskuter forskellige muligheder for at forbedre køretiden for grænseværdiberegningen. ■

## 9 Gode råd

Tidspunkter for opgavehjælp er annonceret på kursets hjemmeside. Desuden vil svar på spørgsmål af generel interesse løbende blive lagt ud på kursets hjemmeside.

Nyhedsgruppen `diku.dat2a` er beregnet som elektronisk diskussionsforum for problemstillinger knyttet til kurset. Instruktorer læser indlæg i denne gruppe regelmæssigt, og bidrager gerne med opklaring af spørgsmål m.v. Brug nyhedsgruppen.

Grafklassen i LEDA er meget nyttig, men skal anvendes med forsigtighed. Specielt skal grafer altid parameteroverføres ved reference, idet en graf og dens kopi ikke indeholder de “samme” knuder — se det udleverede indlæsningsprogram. Det er smart at “skjule” kanter ved at benytte `G.hide_edge()`, men bemærk at `forall_edges` *ikke* gennemløber skjulte kanter. Det er en fordel at benytte typen `edge_array<bool>` til at huske den bedste løsning.

## Litteratur

- [1] T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein: Introduction to Algorithms (2nd ed.), MIT Press, 2001.