Raytracing med simplexalgoritmen

Niels Möller <nisse@lysator.liu.se>

4 juni 1996

Innehåll

Bakgrund

Fredrik Hübinette <profezzorn@lysator.liu.se> och jag pratade för ett tag sen om raytracing. Hubbe hade en idé om att göra en förprocessor till en raytracer/doomklon; tanken är att utgående från en lista av trianglar dela upp rummet i ett antal konvexa polyedrar, och tabellera dessa. För att göra den uppdelningen behöver man bland annat lösa problemet att avgöra om en triangel skär en polyeder, avgränsad av plan. Och just den sortens konvexa polyedrar är simplexalgoritmen bra på att hantera.

Plan representeras av sin normalvektor $\overline{n}$ och sitt avstånd c till origo:

$$P = \{\overline{r} = (x, y, z) : \overline{r} \cdot \overline{n} = c\} \tag{1}$$

Ett halvrum representeras på samma sätt, men med likheten ersatt av en olikhet. En konvex polyeder ges av snittet mellan att antal sådana halvrum. För att det ska vara enkelt att använda simplexalgoritmen krävs dessutom att $x, y, z, c \geq 0$. Att $|n| = 1$ är däremot inte viktigt.

En triangel representeras av det plan som den ligger i, tillsammans med ytterligare tre plan som avgränsar triangeln. Med andra ord, som snittet mellan ett plan och tre halvrum.

Givet ett antal trianglar och ett antal konvexa polyedrar, så kan man göra så här (i pseudopythonkod)

```
for t in triangles:
   for p in polyeders:
      if intersectp(p, t):
         p1, p2 = split(p, t)
         polyeders.replace(p, p1, p2)
```

Här betyder `split` att låta det plan som triangeln ligger i skära polyderern i två delar, vilket är rätt enkelt. Det kluriga är `intersectp`, att avgöra om en triangel skär en polyeder eller inte. Den här artikeln försöker reda ut hur detta problem kan lösas med hjälp av simplexalgoritmen.

Simplexanpassad problemformulering

Vi vill undersöka snittet mellan en triangel och en konvex polyeder. Snittet ges av lösningsmängden till

$$\begin{aligned} \overline{r} \cdot \overline{n} &= c && (2)\\ \overline{r} \cdot \overline{a_i} &\leq \alpha_i \text{ , för } 1 \leq i \leq k && (3)\\ \overline{r} \cdot \overline{b_i} &\geq \beta_i \text{ , för } 1 \leq i \leq l && (4) \end{aligned}$$

Här ska samtliga $c, \alpha_i, \beta_i \geq 0$. Ekvation (2) beskriver planet som triangeln ligger i, medan olikheterna beskriver både polyedern och avgränsningen av triangeln.

Första steget är att införa så kallade *slackvariabler*, för att kunna ersätta olikheterna med likheter. Då ersätts ekvationerna (3) och (4) av

$$\begin{aligned} \overline{r} \cdot \overline{a_i} + s_i &= \alpha_i \text{ , för } 1 \leq i \leq k && (5)\\ \overline{r} \cdot \overline{b_i} - r_i &= \beta_i \text{ , för } 1 \leq i \leq l && (6) \end{aligned}$$

Även $r_i, s_i \geq 0$.

Vi har nu $m = 1 + k + l$ ekvationer, i de $3 + k + l$ variablerna $\{x, y, z, s_i, r_i\}$. I simplexalgoritmen hanterar man *tillåtna baslösningar*, vilket betyder att man sätter alla varibler utom m stycken till 0 (så kallade *icke-basvariabler*) och löser ut de övriga m *basvariablerna* ur ekvationssystemet. För att resultatet ska ligga i det tillåtna området krävs dels att det ekvationssystem man får när man stryker icke-basvariablerna är inverterbart, dels att elementen i lösningsvektorn av basvariabler är icke-negativa.

I vårt ekvationssystem är det mest praktiskt att starta med $x = y = z = 0$ och $s_i = \alpha_i$. Det vill säga, s_i:na är basvariabler, och x, y, x är icke-basvariabler. För att få tillräckligt många behändiga basvariabler inför vi så kallade *artificiella variabler* i ekvationerna (2) och (6). Då får vi ett nytt ekvationssystem,

$$\begin{aligned} \overline{r} \cdot \overline{n} + w_0 &= c && (7) \\ \overline{r} \cdot \overline{a_i} + s_i &= \alpha_i, 1 \leq i \leq k && (8) \\ \overline{r} \cdot \overline{b_i} - r_i + w_i &= \beta_i, 1 \leq i \leq l && (9) \\ x, y, z, s_i, r_i, w_i &\geq 0 && (10) \end{aligned}$$

En lösning till detta ekvationssystem ger en punkt i det snitt vi är intresserade av, om och endast om alla $w_i = 0$. Därför söker vi minimum av

$$f = \sum_0^l w_i \tag{11}$$

Triangeln och polyedern skär varandra om och endast om detta minimum är 0.

Vi har nu $m = 1 + k + l$ ekvationer i $1 + k + 2l$ variabler, och för detta nya problem kan vi enkelt välja en startpunkt: välj som basvariabler $s_i = \alpha_i$ för $1 \leq i \leq k$, $w_0 = c$, $w_i = \beta_i$ för $1 \leq i \leq l$

Innan vi startar simplexalgoritmen måste vi eliminera basvariablerna ur målfunktionen. Om vi löser ut w_i:na, sätter in i (11), och flyttar över konstanterna, får vi

$$f + \overline{r} \cdot (\overline{n} + \sum_1^l \overline{b_i}) - \sum_1^l r_i = c + \sum_1^l \beta_i \tag{12}$$

Hela tiden i simplexalgoritmen har man delat upp variablerna i m basvariabler och $n - m$ icke-basvariabler, vilket svarar mot ett visst hörn i det tillåtna området. Och man eliminerar basvariablerna ur målfunktionen.

För att minska värdet på f i ekvation (12), så tar vi någon av de variabler vars koefficient är positiv, och ökar den. Detta är vår *inkommande* basvariabel. Vi kan öka den ända tills någon av basvariablerna (som inte ingår i målfunktionen, utan bara i bivillkoren) blir 0. De. Vi tar alltså och byter ut en av basvariablerna, på ett sådant sätt att målfunktionens värde minskar. På samma gång så flyttar vi oss längs en kant i det tillåtna området från ett hörn till ett grannhörn. För detaljer, se nästa avsnitt.

Simplexalgoritmen kan stoppa på två sätt

- Ingen inkommande basvariabel: Om ingen av icke-basvariablerna har en positiv koefficient i uttrycket för målfunktionen, så betyder att det inte går att minska målfunktionen ytterligare. I vårt fall så skär triangeln och polyedern varandra om och endast om målfunktionens värde är 0. Men eftersom man i praktiken använder flyttal, så får man i stället använda villkoret $f < \epsilon$.

- Ingen utgående basvariabel: Om den nya basvariabeln kan ökas obegränsat utan att nån av de andra variablerna blir 0, så är målfunktionen nedåt obegränsad. I vårt fall kan det egentligen inte inträffa; målfunktionen $\sum w_i$ kan aldrig bli negativ. Men konstiga avrundningsfel skulle möjligen kunna stoppa simplexalgoritmen här i alla fall.

Simplexalgoritmen

För att lösa ett minimeringsproblem med n variabler och m bivillkor, så använder vi följande data: en n-vektor c, en $m \times n$-matris A, ett högerled b som är en vektor av m icke-negativa tal, och en indexvektor B av m index som beskriver de aktuella basvariablerna. Dessutom behöver vi ett "högerled" till målfunktionen, som vi kallar b_0 (detta tal kan i allmänhet vara negativt, men inte med den målfunktion vi använder). Problemet kan då formuleras

$$\begin{aligned} \min f \text{ då } f + c^T x &= b_0 && (13) \\ Ax &= b && (14) \\ x_i &\geq 0 \text{ , för } 1 \leq i \leq n && (15) \end{aligned}$$

När vi arbetar med problemet vill vi hela tiden ha det på *kanonisk form*, vilket betyder att basvariablerna ska vara eliminerade ur (13), och att motsvarande kolonner i A ska bilda en enhetsmatris. Med andra ord ska $A_{kB_j} = \delta_{kj}$. Man behöver förstås inte spara dessa koloner i A, och i vilket fall så måste man räkna som om elementen där är *exakt* 0 och 1.

Simplexalgoritmen går ut på att genomföra följande tre steg ända tills nåt av stoppvillkoren är uppfyllt.

- Välj ny inkommande basvariabel.
- Välj utgående basvariabel.
- Gör radoperationer i A, b och c tills problemet är på kanonisk form (för den nya uppsättningen basvariabler).

Ny inkommande basvariabler måste väljas bland de som har positiva koefficienter i målfunktionen, det vill säga i c. Men vilken ska man ta om det finns flera? En variant är att helt enkelt ta den som är störst. Men då kan man, om man har riktig otur, råka utför att algoritmen börjar cykla. Vill man försäkra sig mot cykling så kan man i stället ta den av variablerna som har lägst index (kallas *Blands regel*).

Om vi har valt x_r som inkommande basvariabel, så väljer vi som utgående basvariabel den som först blir 0 när vi ökar x_r. Det är bara de basvariabler som har $A_{kr} > 0$ som kommer att minska. Och av dessa så är det den basvariabel x_{B_k} för vilken b_k/A_{kr} är minst som först blir 0. Med andra ord, bestäm det k som uppfyller $A_{kr} > 0$ och minimerar b_k/A_{kr}. Då blir x_{B_k} utgående basvariabel, och vi kommer ihåg det genom att sätta $B_k = r$.

Nu har vi ändrat uppsättningen av basvariabler, genom att sätta $B_k = r$. För att få problemet på kanonisk form så eliminerar vi element ur A på ungefär samma sätt som vid vanlig gausselimination. Först delar vi rad k i A-matrisen med A_{kr}, så att $A_{kr} = 1$. Sen subtraherar vi multipler av denna rad från övriga rader i A, så att resten av elementen i kolonn r blir 0. På samma sätt elimineras c_r ur målfunktionsraden. Och så måste vi förstås göra likadant med högerleden.

Om A, B, c, b, b_0 är de ursprungliga värdena, och A', B', c', b', b'_0 betecknar de nya värdena, så kan en iteration beskrivs så här:

1. Om alla $c_i \leq 0$, stoppa.
2. Välj ett r, med $c_r > 0$ (med eller utan Blands regel).
3. Om alla $A_{jr} \leq 0$, stoppa.
4. Tag minimum av $\{b_j/A_{jr} | A_{jr} > 0\}$, och låt k vara det lägsta index som ger minimum.
5. Sätt $B'_j = B_j$ för $j \neq k$, $B'_k = r$.
6. Sätt $A'_k = A_k/A_{kr}$ och $b'_k = b_k/A_{kr}$. (A_k betecknar här rad k i A).
7. För $j \neq k$, sätt $A'_j = A_j - A_{jr}A_k$ och $b'_j = b_j - A_{jr}b_k$.
8. Sätt $c' = c - c_rA_k$ och $b'_0 = b_0 - c_rb_k$

Det antal steg som behövs för att hitta minimum är oftast mindre än $3m$, där m är antalet bivillkor.

Exempel

Vi vill undersöka om polyedern

$$\begin{aligned} x + y + z &\leq 2 && (16)\\ x, y, z &\geq 0 && (17)\end{aligned}$$

skär triangeln

$$\begin{aligned} x - z &= 1 && (18)\\ y &\geq 1 && (19)\\ x &\leq 3 && (20)\\ -x + y &\leq 1 && (21)\end{aligned}$$

Inför vi slackvariabler i olikheterna får vi ekvationssystemet

$$\begin{aligned} x + y + z + s_1 &= 2 && (22)\\ x - z &= 1 && (23)\\ y - r &= 1 && (24)\\ x + s_2 &= 3 && (25)\\ -x + y + s_3 &= 1 && (26)\\ x, y, z, s_1, s_2, s_3, r &\geq 0 && (27)\end{aligned}$$

Vi saknar basvariabler i ekvationerna (23) och (24), och behöver därför införa två artificiella variabler w_1, w_2.

	x	y	z	s_1	s_2	s_3	r	w_1	w_2	
f	0	0	0	0	0	0	0	−1	−1	0
s_1	1	1	1	1	0	0	0	0	0	2
w_1	1	0	−1	0	0	0	0	1	0	1
w_2	0	1	0	0	0	0	−1	0	1	1
s_2	1	0	0	0	1	0	0	0	0	3
s_3	−1	1	0	0	0	1	0	0	0	1

(28)

Den översta raden ska tolkas $f - w_1 - w_2 = 0$. Den är *inte* på kanonisk form, eftersom basvariablerna w_1, w_2 inte har eliminerats. Men detta görs enkelt genom att addera deras rader till målfunktionen

	x	y	z	s_1	s_2	s_3	r	w_1	w_2	
f	1	1	−1	0	0	0	−1	0	0	2
s_1	1	1	1	1	0	0	0	0	0	2
w_1	1	0	−1	0	0	0	0	1	0	1
w_2	0	1	0	0	0	0	−1	0	1	1
s_2	1	0	0	0	1	0	0	0	0	3
s_3	−1	1	0	0	0	1	0	0	0	1

(29)

Möjliga inkommande basvariabler är x och y. Vi väljer x. Eftersom $b_1/A_{11} = 2, b_2/A_{21} = 1, b_4/A_{41} = 3$, så blir w_1 utgående basvariabel. Den nya tablån blir

	x	y	z	s_1	s_2	s_3	r	w_1	w_2	
f	0	1	0	0	0	0	−1	−1	0	1
s_1	0	1	2	1	0	0	0	−1	0	1
x	1	0	−1	0	0	0	0	1	0	1
w_2	0	1	0	0	0	0	−1	0	1	1
s_2	0	0	1	0	1	0	0	−1	0	2
s_3	0	1	−1	0	0	1	0	1	0	2

(30)

Nu väljer vi y som inkommande basvariabel, och möjliga utgående basvariabler blir s_1, w_2. Eftersom målet är att få ner de artificiella variablerna till 0 så borde man välja w_2 som utgående variabel, men vi leker dum datanmaskin och väljer s_1 eftersom den är först.

	x	y	z	s_1	s_2	s_3	r	w_1	w_2	
f	0	0	−2	−1	0	0	−1	0	0	0
y	0	1	2	1	0	0	0	−1	0	1
x	1	0	−1	0	0	0	0	1	0	1
w_2	0	0	−2	−1	0	0	−1	1	1	0
s_2	0	0	1	0	1	0	0	−1	0	2
s_3	0	0	−3	−1	0	1	0	2	0	1

(31)

Nu är alla $c_i \leq 0$, så vi har kommit fram till minimum. Vi ser, från första raden, att $f = 0$ är minimum, och kan dra slutsatsen att triangeln skär polyedern. Vi kan också läsa ut värdena på alla variabler: $z = s_1 = r = w_1 = 0$ (icke-basvariabler), och $y = 1, x = 1, w_2 = 0, s_2 = 2, s_3 = 1$. Man kan kontrollera att detta verkligen är en lösning till de ursprungliga ekvationssystemet.

Slutsatsen är alltså att polyedern och triangeln skär varandra, och att snittet innehåller punkten $(x, y, z) = (1, 1, 0)$.