Discussie-rubriek

Antwoord aan W.H. van Hoboken.

Hartelijk dank voor de kennelijke zorgvuldigheid waarmee U mijn artikel gelezen hebt. Gaarne ga ik, tot mijn spijt zeer vertraagd, in op de door U gemaakte opmerkingen (MDN, 4, 1977, p. 136-137).

De door U genoemde algoritmes zijn mij inderdaad niet bekend. Helaas hebt U echter verzuimd ze in een literatuuropgave te noemen. Misschien kan dit alsnog gebeuren.

In principe zijn er in mijn algoritme twee stappen:
- het "splitsen" van een ontwerp-clique in twee nieuwe ontwerp-cliques als dit nodig is;
- het elimineren van overbodige ontwerp-cliques.

In het door mij geschreven computerprogramma neemt de eerste stap in totaal (d.w.z. gesommeerd over n-1 rijen; n is het aantal rijen van de matrix T) ongeveer 5% van de rekentijd in beslag. De overige 95% gaat op aan het elimineren van overbodige ontwerp-cliques. Het is dus zaak vooral deze eliminatie zo efficiënt mogelijk te laten verlopen, zoals U ook zelf vaststelt.

Door de opzet van het programma moet de j-de ontwerp-clique in principe vergeleken worden met N_k-j andere ontwerp-cliques (N_k is het totaal aantal ontwerp-cliques aan het begin van de k-de eliminatiestap). Dat zijn in totaal $\sum_{j=1}^{N_k}(N_k-j) = \frac{1}{2}N_k(N_k-1)$ vergelijkingen. Zodra echter de overbodigheid van een ontwerp-clique is gebleken hoeft hij niet meer met verdere ontwerp-cliques vergeleken te worden.

Stel dat gemiddeld slechts een fraktie a_k van de N_k-j ontwerp-cliques de j-de ontwerp-clique hoeft te passeren; dan is het totaal aantal vergelijkingen nog slechts $\frac{1}{2}a_kN_k(N_k-1)$.

Stel vervolgens dat $a_k = m_k/N_k$, waarbij m_k het aantal ontwerp-cliques is dat overblijft na afloop van de k-de eliminatiestap; dan is het totaal aantal vergelijkingen bij de k-de eliminatiestap $\frac{1}{2}(m_k/N_k)N_k(N_k-1) = \frac{1}{2}m_k(N_k-1)$.

Uit de door het programma gegenereerde output blijkt dat onder bovengenoemde assumpties een vergelijking van een ontwerp-clique met een andere steeds gemiddeld .04-.05 msec rekentijd vergde. Deze assumpties (over a_k) lijken dus redelijk.

Dan gaat het erom $\sum_{k=1}^{r} \frac{1}{2} m_k (N_k - 1)$, of, wat bijna op hetzelfde neerkomt, $\sum_{k=1}^{r} m_k N_k$, te minimaliseren (r is het totaal aantal eliminatiestappen). Waarschijnlijk geldt: hoe kleiner r, des te groter N_k; hoe groter r, des te kleiner $(N_k - m_k)$.

Nu ga ik elimineren zodra het aantal ontwerp-cliques na de k-de eliminatiestap groter is geworden dan N_k+difmax. Door te permuteren naar toenemende rijtotalen hoopte ik het aantal eliminatiestappen r te verkleinen; dat is ook gelukt. Bovendien namen door de permutatie de N_k aanzienlijk af, terwijl de verhouding m_k/N_k iets toenam. Vergeleken bij een rekentijd van ca 5 minuten en een maximum aantal ontwerp-cliques van ca 4000 bij de ongepermuteerde matrix T zijn de in "Example" genoemde getallen zeker een vooruitgang!

Uw konstatering dat het versnellingseffekt van permutatie alleen opgaat voor permutatie naar aflopende rijtotalen moet dus haast wel programma-gebonden zijn, dan wel afhangen van de door U voor difmax gekozen waarde van oneindig (helaas is het uit Uw opmerkingen niet duidelijk of de vermelde rekentijden inclusief of exclusief de generatie- en permutatiestap zijn; evenmin, trouwens, waarom de in Uw eerste kanttekening genoemde algoritmen "efficiënter" zijn dan het door mij beschrevene).

Overigens kan het best zijn dat permutatie naar aflopende rijtotalen nog gunstiger is dan die naar oplopende rijtotalen, zoals Uw data suggereren (hoewel ook hier dezelfde onduidelijkheid als voornoemd). Ik heb dat met mijn programma door omstandigheden niet kunnen nagaan. Een probleem is dat U met difmax = ∞ hebt gewerkt (dus r=1), en dus veel ontwerp-cliques hebt willen accepteren. Dit maakt een vergelijking inderdaad erg moeilijk.

Wat vindt U overigens van de "different approach" ?

Met vriendelijke groeten,

F.W. Wilmink