Generaliseren van vlakkenpartities (2)

GAP-trees, testresultaten en verbeteringen

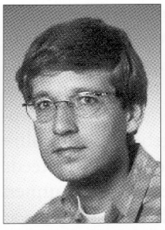
geo-information engineering, spatial data structures, theory | KEYWORDS
geo-informatievoorziening, ruimtelijke gegevensstructuren, theorie | TREFWOORDEN

Zijn in het eerste deel van dit artikel in het oktobernummer de theorie en de implementatie beschreven, dit tweede deel gaat nader in op de resultaten van het testen van de GAP-tree bij het gebruik van een vlakGBKN en de Top10vector. Ook zullen twee belangrijke verbeteringen worden beschreven, namelijk parallelle lijnen om lineaire objecten beter te kunnen meenemen in het generalisatieproces, onder andere door deze te overdrijven, en een aanpassing om met zeer grote datasets te kunnen omgaan.

Twee verschillende grootschalige topografische datasets zijn gebruikt voor het testen van de GAP-tree: de GBKN, meestal in beheer bij het Kadaster, en de Top10vector van de Topografische Dienst Nederland. Beide geografische datasets zijn een nauwkeurige weergave van de terreinsituatie zonder verplaatsingen van objecten vanwege kartografische redenen. Wel zijn in de Top10vector sommige objecten vereenvoudigd weergegeven.

Resultaten met de GBKN

De GBKN is in de meeste gevallen vervaardigd via stereo-fotogrammetrie, aangevuld door naverkenning in het terrein, vooral wat de gebouwen betreft. De presentatieschaal is 1:1.000 (1:2.000 in landelijk gebied) en de dekking is landelijk. De inhoud bestaat uit gebouwen, wegen, waterwegen, spoorwegen en terrein. De nauwkeurigheid is beschreven in termen van relatieve precisie: in landelijk gebied is de relatieve precisie van de afstand tussen twee goed gedefinieerde punten beter dan $40\sqrt{2}$ cm en in bebouwde gebieden is dit beter dan $20\sqrt{2}$ cm. De GBKN is een lijngebaseerde kaart zonder expliciete vlakobjecten. In de toekomst wordt deze wellicht toch topologisch gestructureerd en zullen vlakobjecten wel expliciet worden gerepresenteerd in de zogenaamde objectgerichte of vlakGBKN. Een prototype vlakGBKN is vervaardigd van het gebied rond Zevenaar en bevat zes hoofdtypen vlakobjecten (tabel 1).



drs. J.D. van Putten[1]), Nationaal Lucht- en Ruimtevaartlaboratorium te Marknesse, en prof. dr. ir. P.J.M. van Oosterom, TU Delft, afdeling Geodesie, sectie GISt.

[1]) Dit onderzoek werd uitgevoerd tijdens de periode dat de auteur bij het Kadaster meewerkte aan dit project voor het behalen van haar doctoraalexamen.

De generalisatie-eisen voor de vlakGBKN hangen af van de doelstelling van de kaart. Stel dat dit het weergeven is van stedelijke gebieden en de verbindende wegen. Kruisingen krijgen een lage prioriteit en mogen aan de wegen worden toegevoegd. Neem verder aan dat spoorwegen en water(wegen) ook niet zo belangrijk zijn en al snel aan het terrein mogen worden toegevoegd. In de bebouwde gebieden zijn er de volgende richtlijnen:

- terreinen geleidelijk aan de gebouwen toevoegen;
- vervolgens wegen aan de vergrote gebouwen toevoegen (eerst de onbelangrijke wegen, daarna ook de belangrijkere wegen);
- uiteindelijk zou er slechts een bebouwd gebied moeten overblijven en eventueel nog een aantal belangrijke wegen.

Tabel 1. priotiteit_t(a): GBKN objectklasse prioriteit.

Naam	kleur	code	prioriteit
Terrein	groen	TRN	5
Weg	grijs	WEG	80
Water	blauw	WTR	10
Kruising	zwart	KNP	2
Gebouw	rood	GBW	400
Spoorbaan	bruin	SBN	20

In landelijk gebied zijn er de volgende uitgangspunten:
- gebouwen mogen aan terrein worden toegevoegd;
- verschillende terreinen (en andere soorten vlakken, spoorwegen, waterwegen) mogen worden samengenomen;
- wegen moeten zo lang mogelijk behouden blijven;
- uiteindelijk zou er slechts één groot terrein moeten overblijven.

In de GBKN is er geen onderscheid tussen de terreinen in stedelijk gebied en de terreinen in landelijk gebied. Het enige verschil is de omvang van deze terreinen. Met dit in het achterhoofd en verschillende iteraties met experimenten zijn de waarden voor de tabellen *prioriteit_t(a)* en *compatibiliteit_t(a,b)* gevuld zoals aangegeven in de respectievelijke tabellen 1 en 2; meer details kunnen gevonden worden in [6]. De visuele resultaten van het on-the-fly generaliseren van de vlakGBKN zijn weergegeven (op gelijke presentatieschaal om de verschillen goed te kunnen zien) in fig. 2 t/m 5 door steeds de minimumvereiste importantie te verhogen. De precieze relatie tussen de minimumvereiste importantie en de geschikte presentatieschaal moet nog worden vastgesteld via aanvullende experimenten. Door een extreem hoge minimale importantie te eisen kan de kaartimpressie nogal veranderen (fig. 5). Op de doelschaal zou dit gebied echter overeenkomen met slechts vier of negen pixels op het beeldscherm, zodat het effect niet storend zal zijn.

Daar de nadruk ligt op het generaliseren van vlakkenpartities met de GAP-tree, zijn lijn- en puntobjecten niet weergegeven. In fig. 4 zijn de belangrijkere wegen in het landelijk gebied niet volledig aanwezig. De enige manier om dit op te lossen is de prioriteit van de wegen te verhogen. Echter de verhoogde prioriteit van wegen zorgt ervoor dat wegen belangrijker worden dan bebouwde gebieden. Eigenlijk zouden de wegen in landelijk gebied geleidelijk aan steeds belangrijker moeten worden. Dit zou gerealiseerd kunnen worden door de wegen steeds iets breder te maken, wat weer inhoudt dat ze iets belangrijker worden. Dit idee wordt verder uitgewerkt in de paragraaf Verbeteringen.

Fig. 3. VlakGBKN 2.

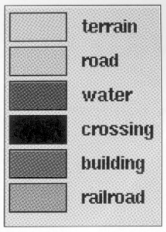


Fig. 1. Legenda van de kaarten.

Resultaten met de Top10vector

De Top10vector is in de meeste gevallen vervaardigd met behulp van monofotogrammetrie, aangevuld door naverkenning in het veld. De presentatieschaal is 1:10.000 en de dekking is inmiddels ook landelijk. Er zijn meer soorten objectklassen voor de vlakobjecten dan in de GBKN, met name wat betreft de verschillende typen terrein. De nauwkeurigheid van de Top10vector is beschreven in termen van absolute precisie in het RD-stelstel: de positie van de punten zou niet meer dan twee meter mogen afwijken. Er wordt voor de Top10vector van hetzelfde generalisatiedoel uitgegaan als voor de GBKN.

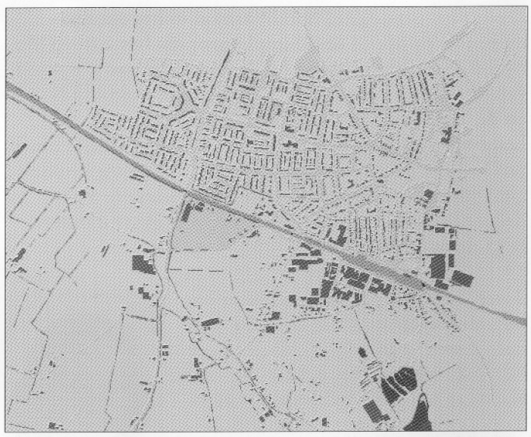


Fig. 2. VlakGBKN 1 (oorspronkelijke kaart).

De Top10vector heeft een aparte classificatie voor terreinen in stedelijk gebied. Deze kunnen dan ook relatief gemakkelijk aan de gebouwen worden toegevoegd door ze een lage prioriteit te geven in verhouding tot de gebouwen. Hierdoor kunnen andere objectklassen weer een relatief hogere prioriteit krijgen in vergelijking met de GBKN. Wegen zijn ook in een aantal categorieën ingedeeld: belangrijk, minder belangrijk en fietspaden. Dit geeft de mogelijkheid om deze verschillende categorieën ook anders te behandelen. Belangrijke wegen kunnen veel langer gehandhaafd blijven dan onbelangrijke wegen en fietspaden door ze een hogere prioriteit te geven. Verder is het wenselijk dat de verschillende losse onderdelen van belangrijke wegen geleidelijk aan samensmelten tot een geheel. Daarom moet de compatibiliteit tussen belangrijke

Fig. 4. VlakGBKN 3.

wegen en andere objectklassen relatief laag worden gehouden, zodat het waarschijnlijker is dat belangrijke wegen aan elkaar worden geplakt. In de GBKN kon dit slechts in beperkte mate worden gerealiseerd, omdat onbelangrijke wegen en fietspaden ook gelijktijdig 'meekomen' (één categorie met belangrijke wegen). In de Top10-vector krijgen de fietspaden de laagste prioriteit van alle typen wegen, maar wel een hoge compatibiliteit met de andere typen wegen. Hierdoor zullen fietspaden sneller dan de andere typen wegen worden verwijderd en hieraan worden toegevoegd.

Na een aantal pogingen en testen van verschillende Top10vector-vullingen voor tabellen *prioriteit_t(a)* en *compatibiliteit_t(a,b)* worden visuele resultaten met de GAP-tree generalisatie behaald, zoals weergegeven in fig. 6 t/m 9. Merk op dat ook op sterk gegeneraliseerde representaties de belangrijke wegen nog steeds aanwezig zijn. Echter, ook voor de Top10vector geldt dat deze gebaat zou zijn bij een aantal verbeteringen in de GAP-tree methode.

Tabel 2. compatibiliteit_t (a,b): GBKN objectklassen compatibiliteit (verticaal: face die wordt verwijderd, horizontaal: face die wordt uitgebreid).

	TRN	WEG	WTR	KNP	GBW	SBN
TRN	0.1	0.1	0.4	0.1	0.7	0.1
WEG	0.2	1.0	0.1	0.9	0.2	0.005
WTR	0.6	0.005	1.0	0.005	0.005	0.005
KNP	0.5	0.9	0.1	1.0	0.005	0.8
GBW	0.9	0.005	0.1	0.005	1.0	0.005
SBN	0.5	0.005	0.1	0.8	0.005	1.0

Verbeteringen in de GAP-tree

Tot nu toe zijn steeds resultaten getoond die met de standaard GAP-tree verkregen zijn. In deze sectie zullen twee belangrijke verbeteringen worden beschreven: parallelle lijnen om lineaire objecten beter mee te kunnen nemen in het generalisatieproces, onder andere door deze te overdrijven, en een aanpassing om met zeer grote datasets om te kunnen gaan.

Introductie van parallelle lijnen

Hiervoor is aangegeven dat de breedte van wegen geleidelijk aan vergroot dient te worden. Dit wordt bereikt door parallelle lijnen aan deze lineaire objecten in de input-datasets toe te voegen. Om de wegen geleidelijk te vergroten moeten één of meer van deze parallelle lijnen op verschillende afstanden voor deze wegen worden gecreëerd, bijvoorbeeld op d, 2d, 4d, enzovoort. Op deze manier liggen de kleinere, minder belangrijke, door de parallelle lijnen ingesloten, stroken dichter bij de originele wegobjecten en zullen ze eerder aan het originele object worden toegevoegd. Het aantal parallelle lijnen en de initiële afstand d zijn parameters voor de objecttypen (gelijk aan de mate van belangrijkheid). Door het één voor één toevoegen van de stroken aan de originele face wordt deze geleidelijk vergroot.

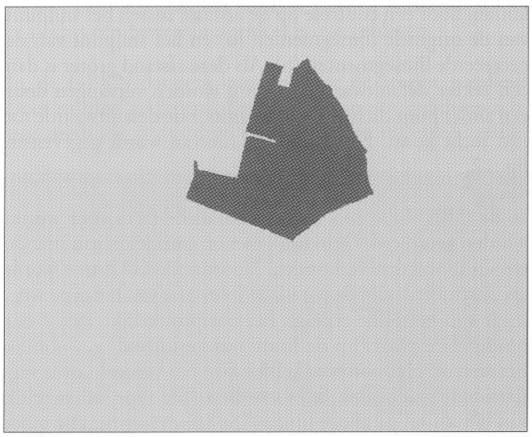


Fig. 5. VlakGBKN 4 (definitieve kaart).

Een manier om een parallelle lijn te construeren is het doortrekken van alle lijnsegmenten van de polylijn (rand). Een parallel lijnsegment kan worden gecreëerd door het bewegen van het originele lijnsegment langs zijn loodlijn. Om van twee aangrenzende parallelle lijnsegmenten een parallelle polylijn te maken dient een snijpunt van de twee parallelle basislijnen te worden berekend. Fig. 10 geeft een voorbeeld van een parallelle polylijn, die op deze manier is gecreëerd. Dit kan voor alle lijnsementen worden gerealiseerd, omdat twee niet-parallelle lijnen altijd een gemeenschappelijk snijpunt hebben (let op de 'uitzonderingen').

Als er twee lijnsegmenten zijn die een heel kleine hoek delen, dan zullen de twee parallelle lijnen een snijpunt hebben dat heel ver weg ligt (fig. 11). Dit kan worden voor-

Fig. 6.
Top10vector 1 (oorspronkelijke kaart).

komen door een controle op de afstand tussen het snijpunt van de originele lijnsegmenten (o) en het snijpunt van de gecreëerde lijnsegmenten (p). Als deze afstand groter is dan een zekere defaultwaarde, moet p worden vervangen door een ander punt dichter bij o, waardoor de defaultwaarde tot zijn recht komt. Een voorbeeld hiervan wordt gegeven in fig. 12.

In de GBKN-datasets kunnen parallelle faces voor wegen worden gecreëerd. Deze faces dienen geleidelijk aan met de wegen te worden verbonden. Voordat de GAP-tree wordt geconstrueerd, zijn de parallelle faces niet van het type weg, maar van hetzelfde type als het oorspronkelijke face *f*, dat op dezelfde plaats op de kaart was gesitueerd, gewoonlijk het terrein. Het oorspronkelijke face *f* is verdeeld over verscheidene gescheiden faces van hetzelfde type, afhankelijk van het aantal parallelle lijnen. De *Samenvoeg*-functie moet erop toezien dat deze parallelle faces geleidelijk aan worden samengevoegd met de betreffende weg. Dat wil zeggen:

Samenvoeg(p, r) > Samenvoeg(p, a)

waar face *r* een weg is, face *p* (een deel van) de parallelle face is, die gecreëerd is voor weg *r* en waarvoor geldt:

face a = face f - face p

In de GBKN-dataset zal deze face naar alle waarschijnlijkheid een terrein zijn. Een manier om het probleem op te lossen is de compatibiliteit tussen TRN-WEG groter te maken dan TRN-TRN. Aangezien dit niet natuurlijk is, zou een betere oplossing voor het probleem kunnen zijn de *Samenvoeg*-functie op een dusdanige manier aan te passen dat ook de buurprioriteit wordt gebruikt om de *Samenvoeg*-waarde te berekenen. Hiervoor dient de *Samenvoeg*-functie als volgt te worden gedefinieerd:

*Samenvoeg(a, b) = prioriteit_t(b) * compatibiliteit_t(a, b) * Lengte(a, b).*

Als deze *Samenvoeg*-functie in het huidige voorbeeld wordt gebruikt, zullen de parallelle faces aan de weg worden toegevoegd. Maar wat zal er gebeuren met alle andere (niet-parallele) faces? Het gevaar bestaat nu dat alle niet-parallelle faces worden samengevoegd met de meest belangrijk buurface in plaats van met de meest compatibele buurface! Nadere testen zijn hier vereist. Het zou nuttig kunnen zijn onderscheid te maken tussen een 'parallelle' face en een 'gewone' face, voor wat betreft prioriteit- en compatibiliteitswaarden.

GAP-tree voor zeer grote datasets

Tijdens het uitvoeren van de testen werd een 'machinebeperking' ontdekt bij het bouwen van een GAP-tree voor een zeer grote dataset. Het huidige algoritme is gebaseerd op een hoofdgeheugenbenadering, voordat het resultaat opnieuw in de database wordt opgeslagen. Dit heeft tot gevolg dat de

Fig. 7.
Top10vector 2.

Fig. 8. Top10vector 3.

datasets te groot kunnen worden om in één keer door het GAP-tree constructieprogramma te kunnen worden verwerkt. De omvang van de datasets zal zelfs nog meer toenemen wanneer parallelle lijnen van de hiervoor besproken verbetering worden geïntroduceerd. Er is dus een techniek vereist waamee de datasets in kleinere eenheden kunnen worden verwerkt, zonder dat dit het resultaat al te zeer aantast. Het is belangrijk op te merken dat in de database waarin de GAP-tree is opgeslagen, geen significante overhead optreedt.

	GBKN		Top10vector	
#randen	8.700	(70.000)	22.500	(125.000)
#faces	4.700	(95.000)	12.000	(150.000)
#gaps	4.700	(120.000)	12.000	(220.000)

Tabel 3. Aantal elementen per representatie (aantal coördinaten tussen haakjes).

In tabel 3 is weergegeven hoeveel coördinaten in de randen (van de planaire topologie) zijn opgeslagen, hoeveel coördinaten in de polygonen (in een representatie zonder topologie) zouden moeten worden opgeslagen en hoeveel coördinaten in the GAP-tree representatie zijn opgeslagen. De overhead (redundante coördinaten) van de GAP-tree is ongeveer 30 tot 40% meer in vergelijking tot de op polygonen gebaseeerde representatie. Men zou verwachten dat het aantal coördinaten in de op polygonen gebaseerde representatie bijna twee keer zo groot zou zijn dan het aantal coördinaten in de op topologie gebaseerde randrepresentatie. Uit tabel 3 blijkt dat dit niet het geval is. Een verklaring kan worden gevonden in het feit dat deze randen een relatief klein aantal punten per lijn hebben en dat het begin- en eindpunt zijn inbegrepen (redundantie in de knooppunten). Verder bevinden zich veel huizen in de datasets, hetgeen resulteert in veel gaten in de topologische structuur. In dit geval heeft de op randen gebaseerde representatie zelfs een coördinaat meer dan de op polygonen gebaseerde representatie, omdat die laatste het begin/eindpunt niet herhaalt.

Echter, ten gevolge van de interne datastructuur van het programma is sprake van een significante overhead tijdens de constructie van de GAP-tree. Stel dat een $2^k * 2^k$ bewerkingsgrid gelegd kan worden over het domein met gemiddeld 4n vlakobjecten met hun centroïde binnen een bewerkingsgridcel. Deze 4n vlakobjecten worden met de GAP-tree gegeneraliseerd tot n vlakobjecten. Vervolgens kan de bewerkingsgridcel worden samengevoegd met zijn drie quadtree buren [2]. De grotere bewerkingsgridcel zal opnieuw gemiddeld 4n gegeneraliseerde vlakobjecten bevatten. Dezelfde stappen kunnen worden herhaald, totdat er slechts één bewerkingsgridcel is overgebleven.

In de GBKN-datasets kunnen voor de wegen parallelle faces worden gecreëerd. Een nadeel van een quadtree-achtige methode is dat deze ertoe leidt dat twee zeer gelijk-

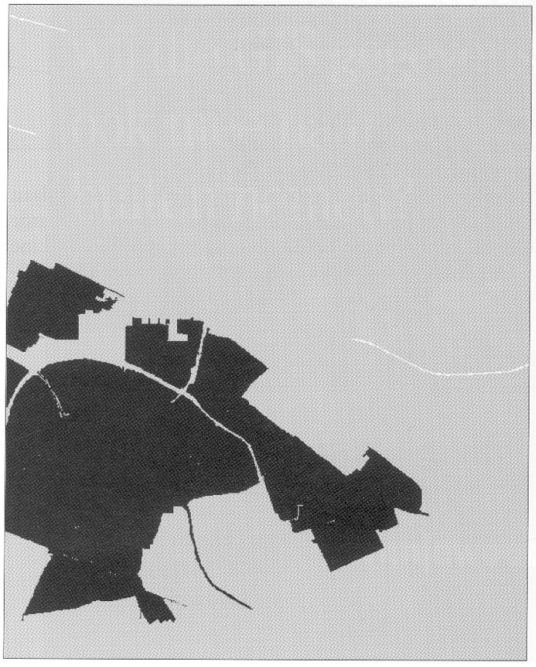


Fig. 9. Top10vector 4 (definitieve kaart).

soortige vlakobjecten, die dicht bij elkaar liggen, maar aan de andere kant van een hoofdquadtree-scheidslijn, niet zullen worden samengevoegd totdat de laatste stap is bereikt. Daarom wordt een Fieldtree-achtige [1] methode voorgesteld, zodat de lijnen die de ruimte verdelen, steeds verschoven zijn op elk niveau in deze structuur. Een bijkomend voordeel van deze 'verdeel en heers'-methode om de GAP-tree te bouwen, is dat het makkelijker is om ter plekke updates te maken zonder dat de hele dataset opnieuw gecomputeriseerd moet worden. De plaatselijke verandering zal de bewerking van de 'buur'gridcel niet beïnvloeden, hoewel alle bovenliggende niveaus wel opnieuw moeten worden berekend.

Conclusie

De GAP-tree is een structuur waarmee alle generalisaties op een dusdanige wijze worden opgeslagen dat voor elke schaal de passende generalisatie snel kan worden bepaald. In dit project werden de GAP-trees voor twee verschillende datasets gebouwd: vlakGBKN en Top10vector. Na het bepalen van de objectklasseprioriteit en de compatibiliteitswaarden, die niet onbelangrijk bleken te zijn, werden de generalisaties volledig automatisch uitgevoerd. In hoeverre aan de generalisatievereisten kan worden voldaan, hangt af van de manier waarop de vlakobjecten kunnen worden onderscheiden. De resultaten met een meer in detail geclassificeerde dataset zijn beter dan met een minder gedetailleerd geclassificeerde dataset.

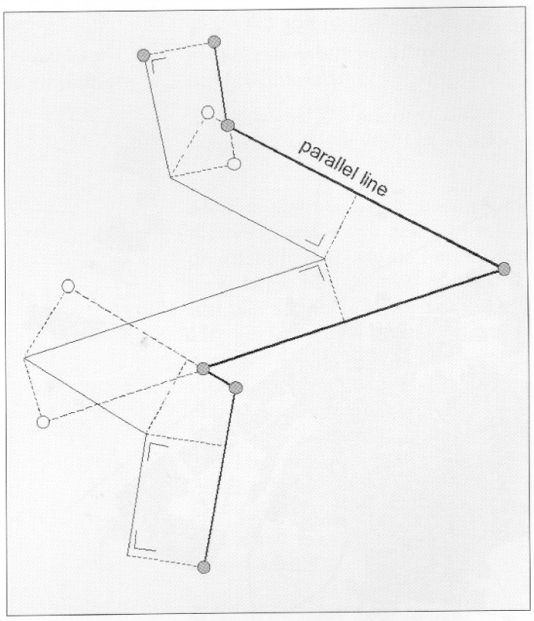


Fig. 10. Parallele lijnen.

Testen met GAP-trees, met parallelle lijnen voor bepaalde objectklassen, moeten nog plaatsvinden. Beter gereedschap om de belangrijkheid van een objectklasse en de compatibiliteit tussen de objectklassen te bepalen zijn noodzakelijk om GAP-trees ook voor andere taken op eenvoudige wijze te kunnen toepassen. Andere datastructuren kunnen worden gebruikt in combinatie met de GAP-tree. De Reactive-tree kan worden gebruikt voor het efficiënt opslaan van de GAP-tree in een DBMS, gebaseerd op de waarde van de locatie en de belangrijkheid [4]. De Binary Line Generalization-tree (BLG-tree) [5] zorgt voor het (lijn)-vereenvoudigingsdeel van het generalisatieproces.

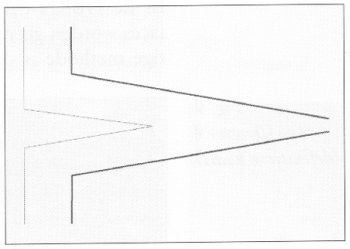


Fig. 11. Situatie met heel kleine hoeken.

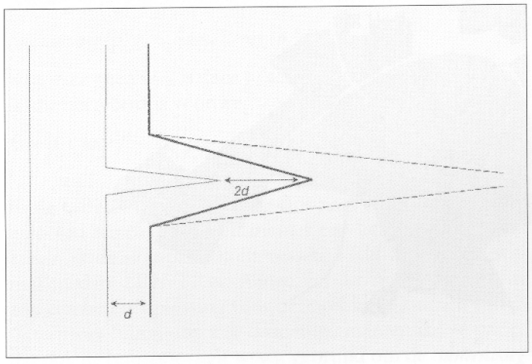


Fig. 12. Oplossing voor een heel kleine hoek.

Verder onderzoek is nodig om te bepalen hoe de GAP-tree op een efficiënte wijze in stand kan worden gehouden tijdens editing-operaties in plaats van het herbouwen van de complete GAP-tree. Een oplossing hiervoor kan de 'verdeel en heers'-methode voor grotere datasets zijn, zoals deze in de vorige paragraaf is voorgesteld. Tenslotte is ook onderzoek vereist naar de mogelijkheden van het opnemen van andere generalisatietechnieken om symbolisering en verplaatsing te ondersteunen [3].

De GAP-tree is erg geschikt bij vectorkaarttoepassingen voor internet. In eerste instantie behoeven voor de overzichtskaart alleen de hogere niveaus van de GAP-tree te worden overgezonden. Deze wordt dan met meer details gevuld tot het moment waarop dit proces, hetzij door de gebruiker, hetzij automatisch, wordt gestopt. Als een gebruiker inzoomt, kunnen knopen op een lager niveau van de GAP-tree worden overgezonden, waardoor meer details worden toegevoegd bovenop de buffer met bovenliggende knopen die al aan de gebruikerskant worden weergegeven.

Summary

Generalization of Area Partitions (2)

The GAP-tree is a structure which can store generalizations in such a way that for every scale the matching generalization can rapidly be determined. In this research GAP-trees were built for two different datasets, the vGBKN and the Top10vector. After setting the feature type priority and compatibility values, the generalizations were performed automatically. This is of great importance for efficient use of cartographic data via the internet. During webbrowsing, a correct generalization for every maplevel is determined and can be shown very fast. In the previous issue of this magazine the authors discussed the theoretical backgrounds and the implementation. In this second and concluding part they deal with the test results and discuss possible improvements.

De auteurs willen graag Harry Uitermark en Frank van Wijngaarden bedanken voor hun bijdrage aan de discussies tijdens het onderzoek.

Literatuur

[1] Andrew U. Frank and Renato Barrera, *The Field-tree: A data structure for Geographic Information System.* Symposium on the Design and Implementation of Large Spatial Databases, Santa Barbara, California, p. 29-44, Springer-Verlag, Berlin, July 1989.

[2] Hanan Samet, *The Design and Analysis of Spatial Data Structures.* Addison-Wesley, Reading, Mass., 1989.

[3] Stuart Shea, K. and Robert B. McMaster, *Cartographic generalization in a digital environment: When and how to generalize.* Auto-Carto 9, p. 56-67, April 1989.

[4] Oosterom, Peter van, *The Reactive-Tree: A storage structure for a seamless, scaleless geographic database.* Auto-Carto 10, p. 393-407, March 1991.

[5] Oosterom, Peter van, and Jan van den Bos, *An object-oriented approach to the design of Geographic Information Systems.* Computers & Graphics, 13(4): 409-418, 1989.

[6] Putten, Judith van, *Experiences with the GAP-tree.* Master's thesis, Utrecht University, Computer Science Department, INF-SCR-97-30, November 1997.