Database search van tijdreeksen, met toepassing in de firma Medtronic

Tom KELLENS

promotor :
Prof. dr. Jan VAN DEN BUSSCHE

Eindverhandeling voorgedragen tot het bekomen van de graad
Master in de informatica afstudeerrichting databases

universiteit hasselt

Abstract

In deze thesis richten we ons op het probleem van similarity search van tijdreeksen. Similarity search kan onderverdeeld worden in twee categorieën, nl. whole matching en subsequence matching. Dit laatste kan beschouwd worden als een veralgemening van whole matching. We zullen in dit werk zien hoe deze veralgemening met behulp van sliding window technieken kan gerealiseerd worden.

Similarity search is in essentie gebaseerd op een afstandsfunctie. Op basis van het soort afstandsfunctie wordt deze thesis in twee delen opgesplitst. Deel I van dit werk behandelt similarity search op basis van de Euclidische metriek. In deel II daarentegen beschouwen we similarity search gebaseerd op de dynamic timewarping afstand.

In het eerste deel behandelen we twee methoden voor similarity search, nl. de GEMINI-methode en de minimale variantie matching (MVM) methode. Beide methoden werden van een implementatie voorzien en werden hierna uitvoerig getest. Bij al het testwerk in deze thesis hebben we ons specifiek gericht op tijdreeksen afkomstig van ECG-sequenties.

Het tweede deel hebben we aangevat door de dynamic timewarping afstandsmeting meer in detail te bekijken. Vervolgens hebben we een methode voor similarity search van tijdreeksen, gebaseerd op deze afstandsmeting, behandeld. Ook deze methode werd ten slotte geïmplementeerd en uitvoerig getest.

Voorwoord

Ter voltooiing van mijn master opleiding Informatica aan de transnationale Universiteit Limburg heb ik deze thesis over database search van tijdreeksen gerealiseerd. Gezien mijn interesse in databases wist dit onderwerp meteen mijn aandacht te trekken. De mogelijkheid om al een klein beetje de werksfeer te kunnen opsnuiven in een groot bedrijf, heeft tevens een aandeel gehad in het maken van mijn keuze.

Het bewerkstelligen van deze thesis heeft behoorlijk wat inspanningen gevergd. Het zou me echter nooit gelukt zijn zonder de steun en de hulp van anderen. Ik zou hiervoor dan ook graag enkele personen willen bedanken. Eerst en vooral zou ik mijn dank willen overbrengen aan mijn promotor en begeleider Prof. dr. Jan Van den Bussche. Zowel zijn suggesties als zijn antwoorden op mijn vele vragen, zijn van groot belang geweest bij de realisatie van dit werk. Daarnaast zou ik ook Richard Houben en Tim Jongen van Medtronic BRC [1] willen bedanken. Ondanks hun drukbezette job wisten ze steeds de nodige tijd voor mij vrij te maken. Ook Nele Schoups verdient een pluim voor haar steun en het meermaals nalezen van mijn thesistekst. Uiteraard mag ik ook mijn medestudenten niet vergeten.

Tot slot zou ik ook graag mijn ouders willen bedanken voor hun steun en hulp gedurende mijn studies. Tevens wil ik hun langs deze weg laten weten dat ik hun enorm dankbaar ben voor de vele opofferingen die ze voor mij gedaan hebben.

[1] http://www.medtronic.com/NL

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Hoofdstuk 1

Inleiding

Een tijdreeks is een begrip dat sommigen onder ons vreemd in de oren zal klinken. Nochtans zijn tijdreeksen alomtegenwoordig. Waarschijnlijk is ieder van ons, zonder het zelf te beseffen, er reeds mee in contact gekomen. We kunnen een *tijdreeks* beschouwen als een opeenvolging van reële getallen, waarmee het verloop van een bepaalde grootheid in de tijd beschreven wordt. Dergelijke data worden dus gekenmerkt door de aanwezigheid van de tijdscomponent. Voorbeelden van tijdreeksen zijn in ontelbare domeinen terug te vinden, zoals bijvoorbeeld:

- Economie: maandelijkse winsten, geldkoersen, ...
- Meteorologie: dagelijkse regenval, temperatuurschommelingen, ...
- Geneeskunde: metingen van hersengolven, hartslagmetingen, ...

Mede door de talloze ontwikkelingen van elektronische meetapparatuur en allerhande automatiseringen, is het aanbod aan tijdreeksen de voorbije decennia exponentieel toegenomen. Vanuit vele onderzoeksdomeinen weerklonk dan ook de roep naar methoden om tijdreeksen zo efficiënt en zo nauwkeurig mogelijk te analyseren, te vergelijken,... Met het oog hierop is de link naar databasesystemen en -applicaties vlug gelegd. Tijdreeksen zijn bijgevolg de laatste jaren hoe langer hoe meer prominent aanwezig in databases.

Op het vlak van toepassingen met betrekking tot tijdreeksen vormt er zich een brede waaier aan mogelijkheden. Het maken van voorspellingen gebaseerd op tijdreeksen is hier een voorbeeld van. Een ander voorbeeld is het lokaliseren van een specifiek patroon in een tijdreeks. Bij het overgrote deel van dergelijke toepassingen komt het er hoofdzakelijk op neer om tijdreeksen onderling met elkaar te vergelijken. Op het vlak van tijdreeksendatabases

wordt er hier dan ook heel wat studie naar verricht. Het vergelijken van tijdreeksen op het niveau van databases wordt meestal omschreven als *(database) similarity search* van tijdreeksen.

In deze thesis richten we ons bijgevolg op het aspect van similarity search van tijdreeksen. Dit alles passen we specifiek toe op tijdreeksen afkomstig van een elektrocardiogram of ECG[1].

[1]ECG: zie sectie 2.2

Hoofdstuk 2

ECG

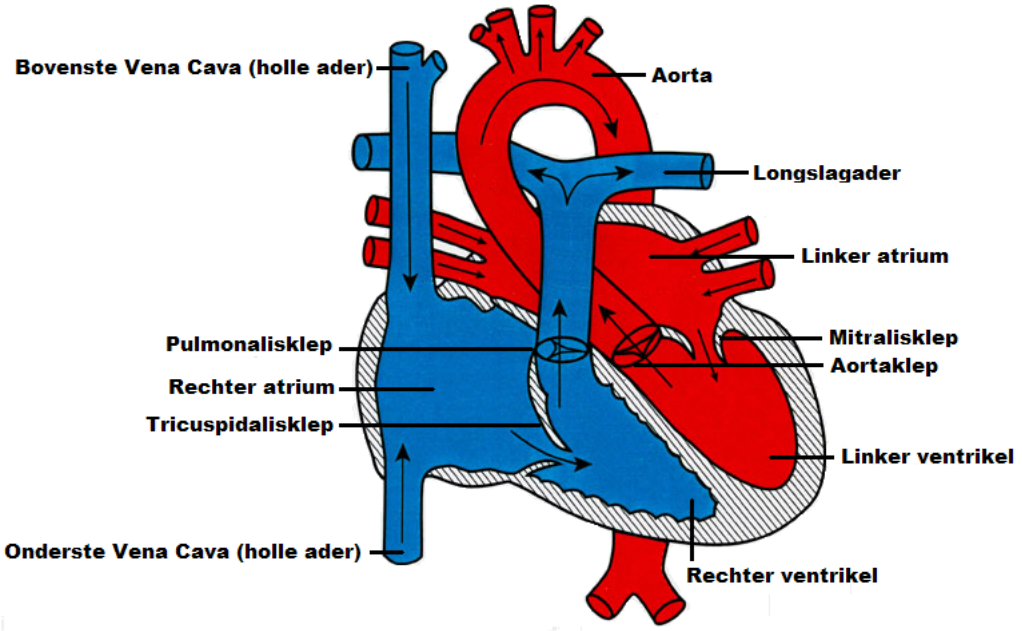
Het opzet van dit hoofdstuk bestaat erin ons van de nodige basiskennis te voorzien van tijdreeksen afkomstig van een elektrocardiogram of kortweg ECG. We vatten dit aan met een korte bespreking van het menselijk hart. Deze bespreking is gebaseerd op de beschrijving in [UZ]. Bij het deel over ECG's baseren we ons op [Yan].

2.1 Het menselijk hart

Het menselijk hart is een holle spier die zorgt voor de circulatie van het bloed doorheen het lichaam. Eigen aan holle spieren is dat ze gevuld zijn met bloed. Onze bloedsomloop kan in twee aparte delen onderverdeeld worden, nl. de kleine en de grote bloedsomloop. Hiervoor is het hart opgebouwd uit een linker en een rechterhelft. De kleine bloedsomloop wordt verzorgd door de rechterhelft van het hart. Hierbij worden de longen via de longslagader bevoorraad met zuurstofarm bloed en wordt het bloed met nieuwe zuurstof verrijkt. Vervolgens komt het bloed in de linkerhelft van het hart terecht. Van hieruit begint de grote bloedsomloop. Deze circulatie voorziet via de aorta de rest van het lichaam van bloed, en zo ook van zuurstof en andere voedingsstoffen. Aan het einde van deze circulatie komt het zuurstofarme bloed weer in de rechter helft van het hart terecht. Vervolgens herhaalt deze cyclus zich, en gelukkig maar!

Beide helften van het hart kunnen in twee onderdelen opgedeeld worden, namelijk het atrium (of boezem) en het ventrikel (of kamer). Beide onderdelen worden via hartkleppen van elkaar gescheiden. Deze hartkleppen zorgen ervoor dat het bloed tijdens het pompen maar in één richting kan gepompt worden. In het rechterdeel van het hart bevinden zich de tricuspidalisklep en de pulmonalisklep. In het linkerdeel vinden we de mitralisklep en de

aortaklep terug. Figuur 2.1 dient ter verduidelijking van de besproken opbouw van het menselijk hart.



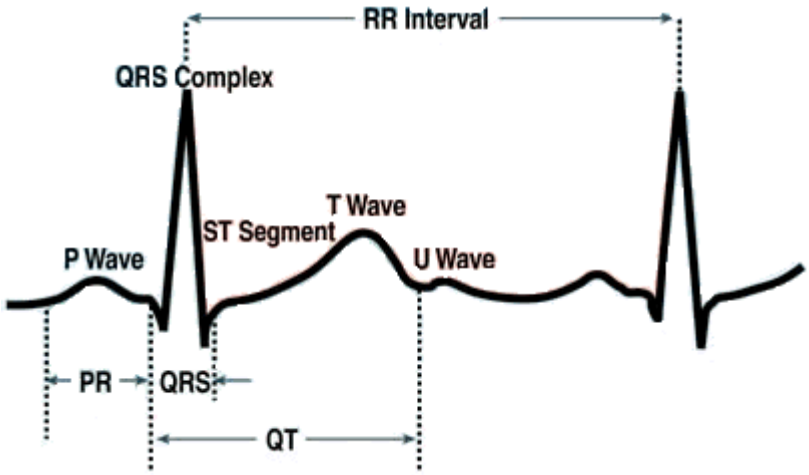
Figuur 2.1: De opbouw van het menselijk hart [UZ]

De pompbeweging van het hart wordt verwezenlijkt door het samentrekken en het ontspannen van de hartspier. Door het samentrekken wordt het aanwezige bloed uit het hart gepompt. In de ontspanningsfase wordt het hart opnieuw met bloed opgevuld. Om dit mechanisme te sturen, is het hart uitgerust met een geleidingsweefsel. Dit weefsel bestaat uit een sino-atriale (SA) knoop, een atrio-ventrikulaire (AV) knoop en verscheidene bundels. Het hart zal dan gestuurd worden door prikkels afkomstig van deze knopen.

2.2 Wat is een ECG?

Een *ECG* of *electrocardiogram* is de registratie van de elektrische activiteit van de hartspier in functie van de tijd. Het samentrekken en het ontspannen van de hartspier leidt namelijk tot elektrische spanningsverschillen in het lichaam. Door deze verschillen op bepaalde plaatsen op het lichaam m.b.v. elektroden te registreren, ontstaat er een ECG. Met behulp van ECG's is het bijvoorbeeld mogelijk om hartritmestoornissen op te sporen.

Een normale hartklopping bestaat uit een opeenvolging van een P-golf, een QRS-complex, een T-golf en tot slot een U-golf. Dergelijke hartklopping wordt geïllustreerd in figuur 2.2. Omdat dit buiten het bereik van dit werk valt, zullen we ons hier niet verder in verdiepen.



Figuur 2.2: Structuur van een normaal ECG [Yan]

2.3 MIT-BIH Arrhythmia Database

We hebben eerder al dat we ons specifiek zullen toeleggen op tijdreeksen afkomstig van ECG's. Dergelijke tijdreeksen zullen we voor de eenvoud benoemen als *ECG-sequenties*, of kortweg *sequenties*. Als testdata voor dit werkstuk maken we gebruik van de ECG's uit de MIT-BIH Arrhythmia Database ([Res]). Deze database vormt een onderdeel van de PhysioBank Database. Hier worden allerlei digitale opnamen van fysiologische signalen en gerelateerde data bewaard voor het gebruik ten dienste van de biomedische onderzoeksgemeenschap.

De sequenties in de MIT-BIH Arrhythmia Database zijn aan 360 Hz gedigitaliseerd. Dit komt neer op 360 samples per seconde. Elke sample heeft hierbij een waarde tussen -5 mV en $+5$ mV.

Hoofdstuk 3

Similarity search

Similarity search van tijdreeksen kunnen we in spreektaal omschrijven als het doorzoeken van een database naar (deel)sequenties gelijkend op een gegeven patroon. Het opzet van dit hoofdstuk bestaat erin het begrip similarity search, ook wel similarity matching genaamd, en enkele aanverwante basisbegrippen verder uit te diepen en strikter te definiëren.

3.1 Basisbegrippen

Vooraleer we dieper ingaan op het onderwerp similarity search zullen we eerst enkele basisbegrippen van naderbij bekijken. Om de gelijkheid van twee sequenties, bijvoorbeeld de sequenties $A\,(= a_1, \ldots, a_n)$ en $B\,(= b_1, \ldots, b_n)$ te meten, wordt er steevast beroep gedaan op een *afstandsfunctie* $D(A,B)$. Twee sequenties worden als *gelijk* beschouwd als ze slechts maximaal een op voorhand gespecifieerde *tolerantie* ϵ van elkaar verschillen. Er wordt in dat geval dan ook gesproken van een *ϵ-match* tussen de twee sequenties, m.a.w. $D(A,B) \leq \epsilon$. Omdat we bij het vergelijken van twee sequenties kijken naar hun onderling verschil, wordt ook wel eens gesproken van een *ongelijkheidsfunctie* in plaats van een afstandsfunctie.

De keuze van de afstandsfunctie heeft een grote invloed op de opbouw van de techniek met als opzet similarity search. In dit werk zullen we twee veel gebruikte metrieken met het oog op similarity search bestuderen, nl. de *Euclidische* en de *Dynamic Time Warping (DTW) afstand*. Gezien het aanzienlijke verschil tussen een aanpak gebaseerd op de Euclidische afstand enerzijds en op de DTW afstand anderzijds, zullen we beide gevallen van elkaar onderscheiden. Ze zullen respectievelijk in deel I en deel II van dit werk aan bod komen.

We spreken af dat we naar de verzameling van sequenties in de database refereren als de *datasequenties* $S_1, S_2, \ldots, S_N$ waarbij N het aantal sequenties in de database voorstelt. Naar de sequentie, opgegeven om m.b.v. similarity search te vergelijken met de datasequenties, verwijzen we steeds als de *querysequentie* Q. De notatie van de juist beschreven begrippen blijven we gebruiken doorheen het verdere vervolg van dit werkstuk, tenzij het anders vermeld wordt. Tabel 3.1 geeft een overzicht van deze notatie.

Tabel 3.1: Overzicht algemene notatie

Q	Querysequentie
$S_1, S_2, \ldots, S_N$	De N datasequenties in de database
S_n $(1 \leq n \leq N)$	De n-de sequentie in de database
$D(S_n, Q)$	Afstandsmeting tussen de sequenties S_n en Q
ϵ	Tolerantie
$D(S_n, Q) \leq \epsilon$	ϵ-match tussen de sequenties S_n en Q

3.2 Classificatie

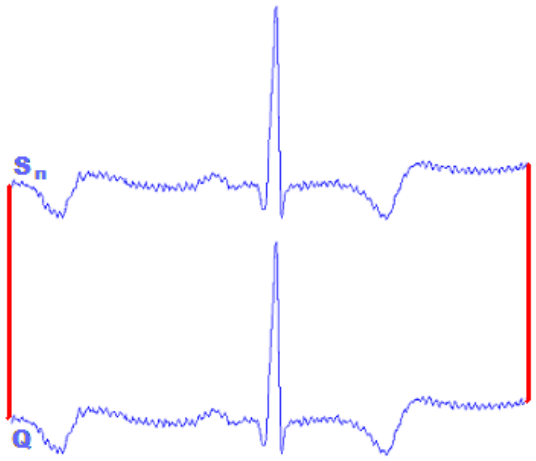
In de bestaande literatuur over dit onderzoeksdomein wordt similarity search steevast opgedeeld in twee categorieën, nl. whole matching en subsequence matching. Deze categorieën worden in [FRM94] en [MWH02] als volgt gedefinieerd:

- **Whole matching**: Gegeven een verzameling van N datasequenties van reële getallen $S_1, S_2, \ldots, S_N$, een querysequentie Q en een tolerantie ϵ zoeken we al deze datasequenties die een ϵ-match met Q vormen. Hierbij moeten zowel de datasequenties als de querysequentie van dezelfde lengte zijn.

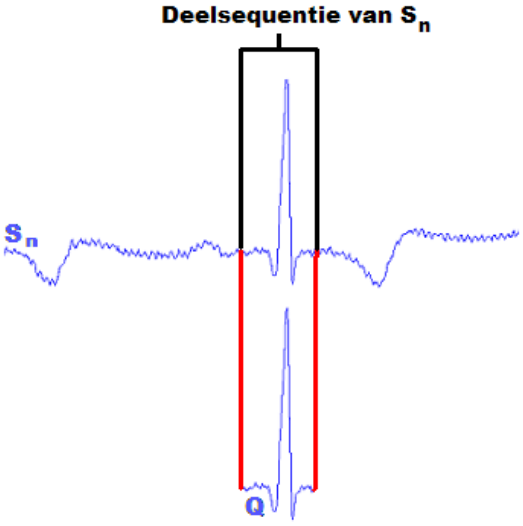
- **Subsequence matching**: Gegeven een verzameling van N datasequenties van reële getallen van willekeurige lengte $S_1, S_2, \ldots, S_N$, een querysequentie Q en een tolerantie ϵ zoeken we al deze datasequenties S_n $(1 \leq n \leq N)$ die één of meer deelsequenties bevatten die een ϵ-match met Q vormen.

In het merendeel van de werkstukken over similarity search van tijdreeksen wordt steeds vanuit het standpunt van whole matching gekeken. Hierbij wordt dan telkens aangehaald dat het subsequence matching probleem

naar het whole matching probleem omgezet kan worden m.b.v. zogenaamde sliding window technieken (zie hoofdstuk 4). Intuïtief kunnen we echter ook vrij eenvoudig inzien dat subsequence matching naar whole matching te herleiden valt. We kunnen namelijk elke mogelijke deelsequentie van een datasequentie S_n $(1 \leq n \leq N)$ m.b.v. whole matching met de querysequentie Q proberen te vergelijken. De figuren 3.1 en 3.2 dienen ter illustratie van begrippen whole en subsequence matching.



Figuur 3.1: Whole matching van de ECG-sequenties S_n en Q.



Figuur 3.2: Subsequence matching van de ECG-sequenties S_n en Q.

3.3 Soorten similarity search query's

Tot slot van dit hoofdstuk behandelen we enkele soorten query's, typisch voorkomend bij similarity search. Met betrekking tot dergelijke query's kunnen we de volgende onderverdeling maken:

- *Range query's*: Gegeven een querysequentie Q, wordt er bij dit soort query's gezocht naar alle data(deel)sequenties die maximaal een tolerantie ϵ verschillen van de querysequentie Q.

- *Nearest neighbor query's*: Gegeven een waarde k ($k \in \mathbb{N}_0$) en een querysequentie Q, zoeken dergelijke query's naar de k data(deel)sequenties, het meest gelijkend op de querysequentie Q.

- *All-pairs query's*: Bij deze query's wordt er gezocht naar alle paren, bestaande uit twee aan elkaar gelijke data(deel)sequenties.

Zowel bij range query's, als bij nearest neighbor query's worden alle data(deel)sequenties steeds vergeleken met een querysequentie. In het geval van all-pairs query's is het echter nodig dat alle data(deel)sequenties onderling vergeleken worden. Omwille van dit laatste gegeven is het dus duidelijk dat all-pairs query's meer kostelijke operaties met zich meebrengen. Het zoeken van k data(deel)sequenties, het meest gelijkend op een querysequentie Q, zouden we ook kunnen verwezenlijken met behulp van range query's i.p.v. nearest neighbor query's. We zouden namelijk eerst kunnen zoeken m.b.v. een range query met een zeer kleine tolerantie ϵ. Indien deze query minder dan k resultaten geeft, herhalen we deze query maar dan met een iets grotere waarde voor ϵ. Dit herhalen we totdat de range query minstens k resultaten weergeeft. Vervolgens zoeken we de k beste matchings tussen de weergegeven resultaten. Om deze redenen zullen we in dit werk onze aandacht voornamelijk vestigen op range query's.

Hoofdstuk 4

Sliding windows

In dit werkstuk zullen we ons volledig richten op het aspect van subsequence matching. In het voorgaande hoofdstuk hebben we gezien dat subsequence matching beschouwd kan worden als een veralgemening van whole matching. We gaan ons nu wat meer toeleggen op de sliding window technieken waarmee we zulke veralgemening kunnen realiseren. Dergelijke technieken staan in voor het splitsen van sequenties in *windows* met het oog op subsequence matching d.m.v. whole matching. Voor het vervolg van deze sectie spreken we de notatie beschreven in tabel 4.1 af.

Tabel 4.1: Overzicht notatie bij sliding windows

$S\ (= s_1, \ldots, s_n)$	Sequentie S bestaande uit n reële getallen, waarbij s_i het i-de reëel getal van S voorstelt
$S\,[i:j]$	Subsequentie van S (van het i-de reëel getal van S t.e.m. het j-de reëel getal van S)
$\|S\|$	Lengte van sequentie S
ω	Grootte van de window

4.1 Subsequence matching m.b.v. sliding windows

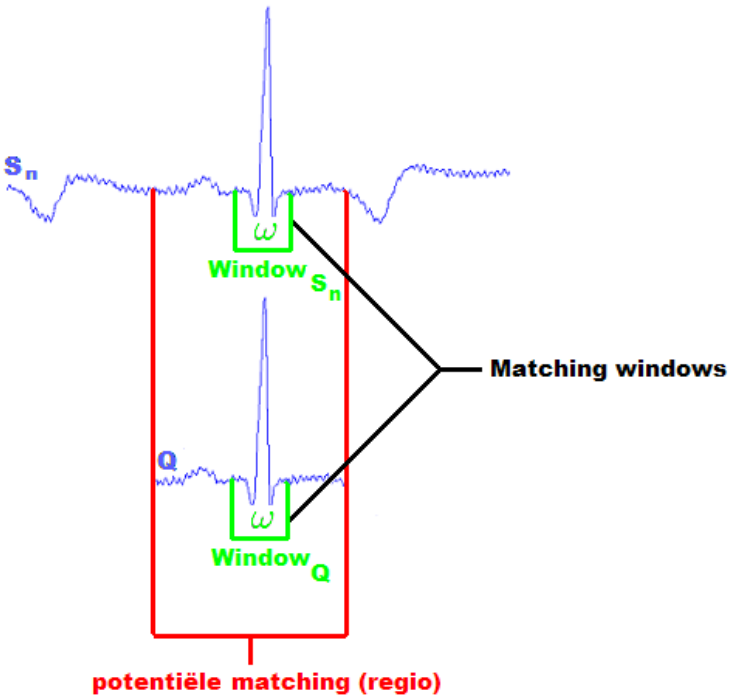
In [FRM94], [MWL01b] en [MWH02] worden enkele technieken voor het opsplitsen van sequenties in windows voorgesteld. Deze technieken verschillen in de wijze waarop ze de datasequenties en de querysequentie in windows van dezelfde lengte verdelen. De algemene werkwijze waarop ze aangewend

worden met als doel subsequence matching is echter steeds identiek. We zullen daarom eerst deze werkwijze behandelen alvorens de sliding window technieken van naderbij te bekijken.

Een deel van het werk kan in een preprocessing-stap uitgevoerd worden. Zo kunnen alle datasequenties op voorhand in windows van lengte ω gesplitst worden. Hoe dit gebeurt en in hoeveel windows de sequenties opgedeeld worden, is afhankelijk van de gebruikte sliding window techniek. Eens dit helemaal voltooid is, kan er naar matchings gezocht worden die overeen komen met een querysequentie Q. Dit gebeurt m.b.v. het volgende meerstappenproces:

- **Stap 1:** De querysequentie Q wordt in windows van lengte ω opgesplitst.
- **Stap 2:** De windows afkomstig van de querysequentie worden m.b.v. een matching-algoritme vergeleken met de windows afkomstig van de datasequenties. Wanneer hierbij een match gevonden wordt, wordt de betrokken datasequentie opgenomen in een *kandidatenverzameling*. Het idee hierachter is dat wanneer een stuk (van lengte ω) van een datasequentie overeenkomt met een stuk (van lengte ω) van de querysequentie, de kans bestaat dat de querysequentie geheel voorkomt in deze datasequentie. De gevonden datasequentie wordt dan m.a.w. als mogelijke kandidaat opgenomen omdat deze misschien de gehele querysequentie bevat.
- **Stap 3:** Deze stap zouden we kunnen beschouwen als een postprocessing-stap. Voor iedere gevonden datasequentie in de kandidatenverzameling wordt m.b.v. hetzelfde matching-algoritme nagegaan of de volledige querysequentie voorkomt in deze datasequentie. Wanneer dit niet het geval is, spreken we van een *false alarm*. Door bij te houden welke window van de gevonden datasequentie overeenkomt met welke window van de querysequentie, kan een bepaalde regio van lengte $|Q|$ in de datasequentie afgebakend worden. De bekomen sequentieregio zou dan een potentiële match met de querysequentie kunnen vormen. Zo is het niet nodig de hele datasequentie nog eens volledig te doorlopen om een mogelijke match te vinden. Enkel de gevonden regio zal dan vergeleken moeten worden met de querysequentie. Deze stap wordt verduidelijkt m.b.v. figuur 4.1.

We zijn er bij de beschrijving van het voorgaande proces van uit gegaan dat we reeds een (whole) matching-algoritme voorhanden hebben. Dergelijke matching-algoritmen zullen in latere secties aan bod komen.



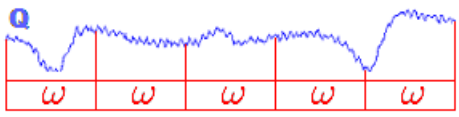
Figuur 4.1: De postprocessing-stap m.b.v. windows

4.2 FRM

FRM is één van de eerste benaderingen tot subsequence matching en werd voorgesteld in [FRM94]. De naam FRM is ontleend aan de beginletter van de drie bezielers erachter, nl. Faloutsos, Ranganathan en Manolopoulos. We bespreken nu de sliding window techniek die hierbij gebruikt wordt.

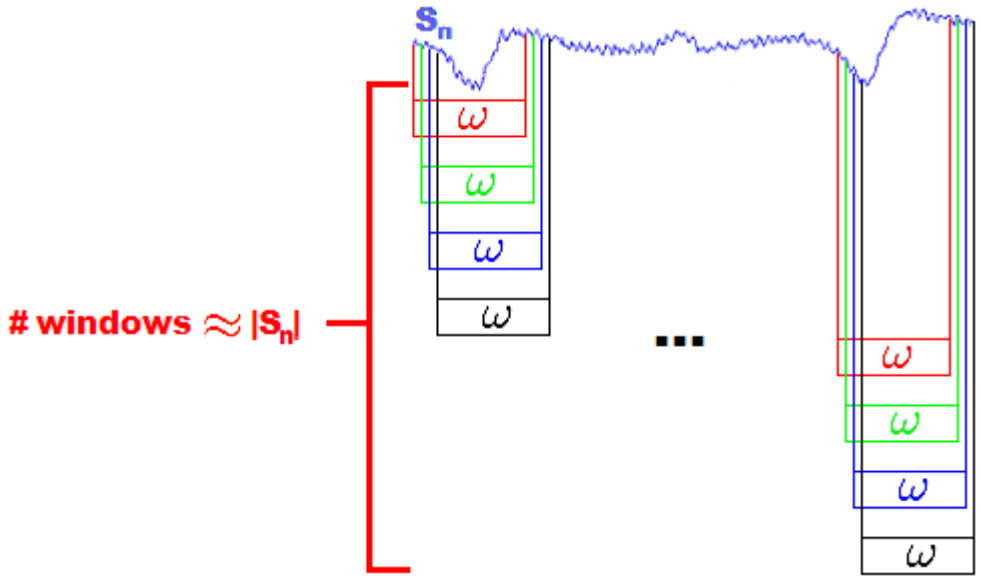
4.2.1 Opsplitsing van de sequenties

Zowel voor de querysequentie, als voor de datasequenties wordt een verschillende splitsingsmethode gebruikt. Voor een querysequentie Q hanteert men een methode waarbij zulke sequentie opgesplitst wordt in stukken van lengte ω. Dit resulteert in $\lfloor |Q|/\omega \rfloor$ aantal deelstukken. Dergelijke stukken noemen we *disjoint windows*. Figuur 4.2 dient ter verduidelijking hiervan.



Figuur 4.2: Het splitsen van een querysequentie Q in disjoint windows

Een datasequentie S_n daarentegen wordt opgesplitst door met een venster van lengte ω over de sequentie heen te 'glijden'. Elke positie van het venster levert ons een window van lengte ω op. Naar de bekomen windows wordt gerefereerd als *sliding windows*. In het totaal worden er op deze manier net iets minder dan $|S_n|$ windows afgesplitst, waarbij S_n de desbetreffende datasequentie voorstelt. We illustreren dit m.b.v. figuur 4.3.



Figuur 4.3: Het splitsen van een datasequentie S_n in sliding windows

4.2.2 Correctheid

In sectie 4.1 hebben we reeds gezien hoe we sliding window technieken kunnen gebruiken met het oog op subsequence matching. Een zeer belangrijk aspect hierbij is het opvullen van de kandidatenverzameling. Hierbij is het van uitermate groot belang dat het aanvullen van dergelijke verzameling niet tot foutieve resultaten kan leiden. Zo mag het namelijk niet voorvallen dat een ϵ-match tussen twee sequenties over het hoofd gezien wordt. Naar het terzijde leggen van zulke match wordt ook wel eens verwezen als een *false dismissal*.

Met als doel de kandidatenverzameling op te vullen, worden al de sliding windows, bekomen uit de datasequenties, vergeleken met al de disjoint windows van de querysequentie. Een datasequentie S_n zal bij FRM tot deze verzameling opgenomen worden als het aan de volgende voorwaarde voldoet: één van de sliding windows van S_n moet matchen met een disjoint window van de querysequentie Q. Tijdens de eerder vermelde postprocessing-stap

zal dan uitgemaakt worden of S_n een deelsequentie bevat die matcht met de querysequentie Q.

De correctheid van FRM is dus grotendeels afhankelijk van de toelatingsvoorwaarde tot de kandidatenverzameling. Het is immers noodzakelijk dat elke datasequentie, die een deelsequentie bevat die matcht met een querysequentie, in deze verzameling terechtkomt. De correctheid van de toelatingsvoorwaarde bij FRM is gebaseerd op de volgende drie lemma's:

Lemma 4.1 ([FRM94]) *Wanneer twee sequenties S en Q van dezelfde lengte opgesplitst worden in p windows s_i en q_i $(1 \leq i \leq p)$ en wanneer S en Q een ϵ-match vormen, dan vormen minstens één van de paren (s_i, q_i) een $\epsilon/\sqrt{p}$-match. M.a.w.*

$$D(S,Q) \leq \epsilon \Rightarrow \bigvee_{i=1}^{p} D(s_i,q_i) \leq \epsilon/\sqrt{p}.$$

Lemma 4.2 ([FRM94]) *Wanneer twee sequenties S en Q van dezelfde lengte een ϵ-match vormen, dan vormt elk paar $(S[i:j], Q[i:j])$ ook een ϵ-match. M.a.w.*

$$D(S,Q) \leq \epsilon \Rightarrow D(S[i:j], Q[i:j]) \leq \epsilon.$$

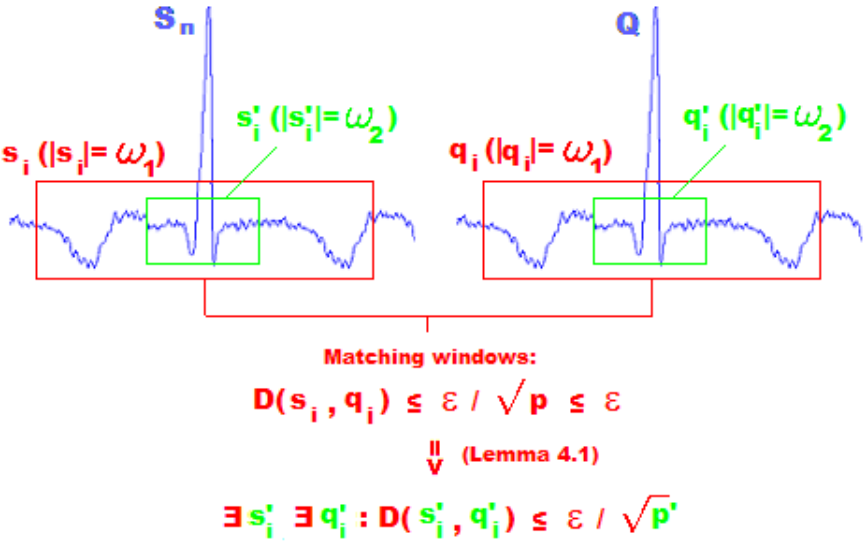
Lemma 4.3 ([FRM94]) *Veronderstel dat een datasequentie S_n opgesplitst wordt in sliding windows van lengte ω en dat een querysequentie Q in disjoint windows van dezelfde lengte opgesplitst wordt. Als een subsequentie $S_n[i:j]$ van lengte $|Q|$ een ϵ-match vormt met Q, dan bestaat er minstens één disjoint window q_k van Q dat een $\epsilon/\sqrt{p}$-match vormt met een sliding window $S_n[i + (k-1) * \omega : i + k * \omega - 1]$ (dat bevat is in $S_n[i:j]$), met $p = \lfloor |Q|/\omega \rfloor$.*

De lemma's 4.1 en 4.2 dienen ter ondersteuning van lemma 4.3. Dit laatste lemma zegt dat, wanneer $S_n[i:j]$ en Q matchen, $S_n[i:j]$ minstens één sliding window moet bevatten die matcht met een disjoint window van Q. We hebben de voorwaarde reeds gezien waaraan S_n moet voldoen om tot de kandidatenverzameling opgenomen te worden. Lemma 4.3 garandeert ons dat er met deze voorwaarde geen false dismissals zullen plaatsvinden. Voor de bewijzen van de drie voorgaande lemma's verwijzen we naar [FRM94].

Met betrekking tot lemma 4.2 is het mogelijk dat een window q_i, afkomstig van de sequentie Q, een $\epsilon/\sqrt{p}$-match vormt met een window s_i, afkomstig van de sequentie S, terwijl Q en S geen ϵ-match vormen. In dergelijke situatie spreken we dus van de eerder vermelde false dismissals.

4.2.3 Maximale windowgrootte

De windowgrootte waarmee gewerkt wordt, heeft een aanzienlijke invloed op het aantal optredende false alarms. Het vergroten van de windowgrootte ω heeft het omgekeerde effect op het aantal false alarms. M.a.w. des te groter ω, des te minder false alarms er voorvallen. Hiernaar wordt verwezen als het *window size effect*. We bekijken dit d.m.v. een voorbeeld waarbij we twee sequenties met windows van zowel grootte ω_1 als ω_2 vergelijken waarbij $\omega_1 > \omega_2$. Wannneer nu een sequentie toegevoegd wordt aan de kandidatenverzameling door te vergelijken met ω_1-windows, zou dit wegens lemma 4.1 ook het geval zijn geweest voor ω_2-windows. Het omgekeerde kan echter niet verondersteld worden. Dit window size effect wordt verduidelijkt m.b.v. figuur 4.4.

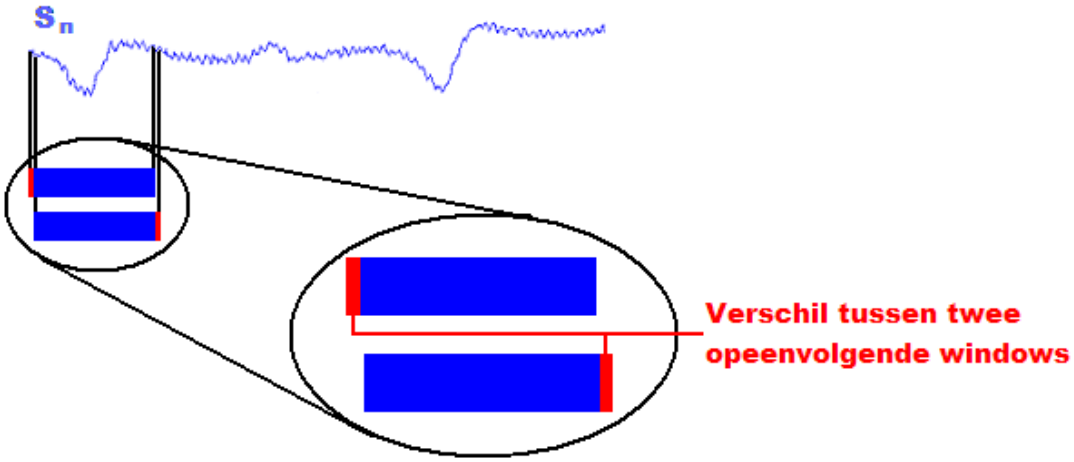


Figuur 4.4: Het windows size effect

Bij FRM is het nodig dat de querysequentie minstens in één window opgesplitst kan worden. De maximale windowgrootte ω is dan ook begrensd door de opgelegde minimale lengte van de querysequentie. Op deze wijze past de kleinst mogelijke querysequentie precies in één window. Alle langere querysequenties worden dan in twee of meer windows verdeeld.

4.3 Dual Match

Bij FRM worden de datasequenties opgesplitst in sliding windows. Vaak is het nodig dat deze sliding windows in de database dienen bewaard te blijven. Zoals in figuur 4.5 te zien is, is het verschil tussen twee opeenvolgende sliding windows van een datasequentie S_n uiterst miniem. Dit leidt uiteraard tot enige vermijdbare 'storage overhead' in de database. Tevens dienen er per datasequentie S_n heel wat ($\approx |S_n|$) sliding windows bijgehouden te worden, dit leidt tot een minder goede search performance. Beide zwakheden van FRM komen nog sterker tot uiting indien er veel en lange datasequenties aanwezig zijn in de database. In [MWL01b] wordt een sliding window techniek, Dual Match genaamd, voorgesteld waarin dergelijke zwakheden uit de weg geruimd worden.



Figuur 4.5: Twee opeenvolgende sliding windows van een datasequentie S_n

4.3.1 Opsplitsing van de sequenties

Net als bij FRM worden de sequenties bij Dual Match opgesplitst in disjoint en sliding windows. Het verschil tussen beide benaderingen schuilt echter in welke sequenties in disjoint en welke in sliding windows opgesplitst worden. We hebben gezien dat bij FRM de datasequenties in sliding windows en de querysequentie in disjoint windows verdeeld worden. Bij Dual Match daarentegen gebeurt dat echter in omgekeerde volgorde. De datasequenties worden er in disjoint windows verdeeld, de querysequentie op haar beurt in sliding windows.

4.3.2 Correctheid

Net als bij FRM is de correctheid van Dual Match afhankelijk van de toelatingsvoorwaarde tot de kandidatenverzameling. Bij Dual Match luidt deze voorwaarde als volgt: er moet minstens één van de disjoint windows van de datasequentie S_n matchen met een sliding window van de querysequentie Q. Ook nu weer zal tijdens een postprocessing-stap uitgemaakt worden of S_n een subsequentie bevat die matcht met de querysequentie Q.

Naast de eerder vermelde lemma's 4.1 en 4.2 steunt de correctheid van Dual Match ook op lemma's 4.4 en 4.5. De bewijzen van de twee laatstgenoemde lemma's zijn respectievelijk terug te vinden in [MWL01a] en [MWL01b].

Lemma 4.4 ([MWL01a],[MWL01b]) *Als een sequentie S opgedeeld wordt in disjoint windows van lengte ω, bekomt men minstens p windows waarbij p bepaald wordt door de volgende formule:*

$$p = \lfloor (|S| + 1) / \omega \rfloor - 1.$$

Lemma 4.5 ([MWL01b],[MWH02]) *Stel dat de datasequentie S_n in disjoint windows van lengte ω en de querysequentie Q in sliding windows van lengte ω opgesplitst worden. Als de subsequentie $S[i:j]$ van lengte $|Q|$ een ϵ-match vormt met Q, dan bestaat er in $S[i:j]$ minstens één disjoint window $S[i+k : i+k+\omega-1]$, dat een $\epsilon/\sqrt{p}$-match vormt met het sliding window $Q[k : k+\omega-1]$. Hierbij geldt: $p = \lfloor (|Q| + 1) / \omega \rfloor - 1$.*

4.3.3 Maximale windowgrootte

De maximale windowgrootte is ook hier weer afhankelijk van de minimale lengte van de querysequentie. Deze onderlinge relatie wordt weergegeven in lemma 4.6. Het bewijs hiervan vindt men terug in [MWL01a].

Lemma 4.6 ([MWL01a],[MWL01b]) *Als $Min(Q)$ de minimumlengte van de querysequentie Q voorstelt, dan wordt de de maximale windowgrootte bij Dual Match weergegeven door $\lfloor (Min(Q) + 1) / 2 \rfloor$.*

Bij de vergelijking van de maximale windowgrootte bij Dual Match blijkt deze slechts half zo groot te zijn als bij FRM. Met betrekking tot het aantal false alarms kunnen we dus wegens het windows size effect stellen dat FRM hiertegen beter gewapend is dan Dual Match.

4.4 General match

Dual Match heeft als voordeel dat het zijn datasequenties opdeelt in disjoint windows. Zo bekomt men opmerkelijk minder windows t.o.v. FRM, dat zijn datasequenties verdeelt in sliding windows. Dit voordeel komt vooral bij databases met veel en lange datasequenties tot zijn recht. FRM daarentegen is op zijn beurt dan weer beter bestand tegen false dismissals dan Dual Match. We behandelen nu een sliding window techniek dewelke de voordelen van FRM en Dual Match combineert. Deze techniek staat bekend als *General Match*.

4.4.1 Opsplitsing van de sequenties

Bij subsequence matching m.b.v. General Match worden de datasequenties opgesplitst in *J-Sliding windows*. De querysequenties daarentegen worden opgedeeld in *J-Disjoint windows*. De begrippen J-Sliding en J-Disjoint windows worden respectievelijk beschreven in de definities 4.1 en 4.2.

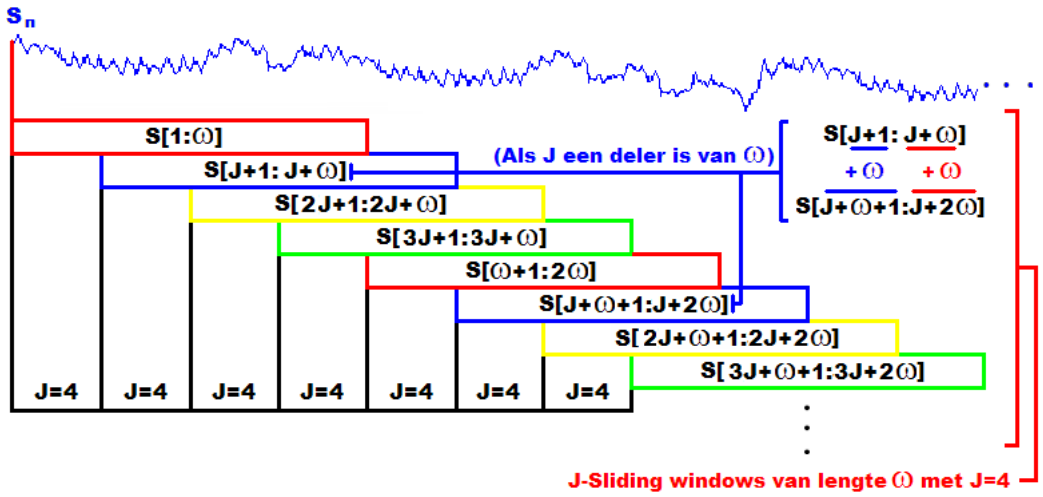
Definitie 4.1 ([MWH02]) *Een* J-Sliding window $(1 \leq J \leq \omega)$ s_i^J *van grootte* ω *van een sequentie* S *wordt gedefinieerd als een deelsequentie van lengte* ω *beginnend vanaf positie* $[(i-1)*J+1]$ *met* $1 \leq i \leq \frac{|S|-\omega}{J}+1$.

Definitie 4.2 ([MWH02]) *Een* J-Disjoint window $(1 \leq J \leq \omega)$ $q_{(i,j)}^J$ *van grootte* ω *van een sequentie* Q *wordt gedefinieerd als een deelsequentie van lengte* ω *beginnend vanaf positie* $[i+(j-1)*\omega]$ *met* $1 \leq i \leq J$ *en* $1 \leq j \leq \frac{|Q|-i+1}{\omega}$.

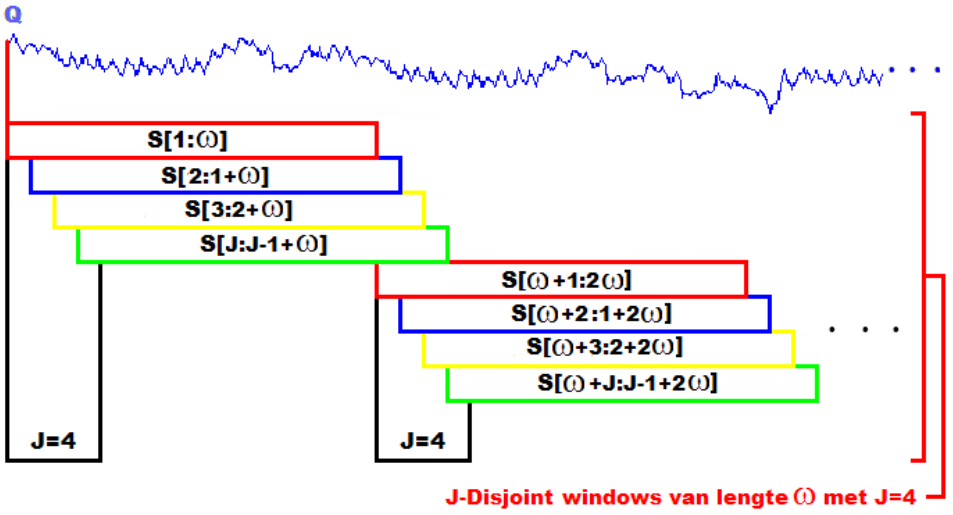
De bovenstaande notaties van J-Sliding en J-Disjoint windows blijven we aanhouden doorheen dit werkstuk en worden voor het overzicht nog eens samengevat in tabel 4.2. Figuur 4.6 toont ons een voorbeeld waarbij een dataysequentie S_n opgedeeld wordt in J(=4)-Sliding windows van lengte ω. In figuur 4.7 wordt de situatie weergegeven waarbij een querysequentie Q verdeeld wordt in J(=4)-Disjoint windows, tevens van lengte ω.

Tabel 4.2: Overzicht notatie bij J-Sliding en J-Disjoint windows

s_i^J	De i-de J-Sliding window van sequentie S, beginnend op positie $[(i-1)*J+1]$ in S met $1 \leq i \leq \frac{\|S\|-\omega}{J}+1$.
$q_{(i,j)}^J$	De (i,j)-de J-Disjoint window van sequentie Q, beginnend op positie $[i+(j-1)*\omega]$ in Q met $1 \leq i \leq J$ en $1 \leq j \leq \frac{\|Q\|-i+1}{\omega}$.



Figuur 4.6: J-Sliding windows [MWH02]



Figuur 4.7: J-Disjoint windows [MWH02]

In de definities 4.2 en 4.1 over J-Sliding en J-Disjoint windows kunnen we zien dat J eender welke waarde tussen één en ω kan aannemen. In het werk [MWH02] wordt er echter aangehaald dat we voor de waarde van J gemakshalve best een deler van windowlengte ω nemen. Een voordeel hiervan is dat wanneer er een J-Sliding window s_a^J in S bestaat met startpositie $[(a-1)*J+1]$, er dan ook een J-Sliding window $s_{a+\frac{\omega}{J}}^J$ zal bestaan beginnend op positie $\left[\left(a+\frac{\omega}{J}-1\right)*J+1\right]$. Dit echter wel in de veronderstelling dat het stuk van sequentie S, gelegen achter window s_a^J, nog minstens ω lang is. Het verschil tussen de beginposities van s_a^J en $s_{a+\frac{\omega}{J}}^J$ is dus precies gelijk aan de lengte van één window, namelijk ω. Deze situatie is ook terug te zien in figuur 4.6. Al de J-Sliding windows $s_{a+n*\frac{\omega}{J}}^J$ $(n \geq 0)$ die zich bevinden in de deelsequentie $S[i:j]$ van een sequentie S staan bekend als de *ingesloten windows* van $S[i:j]$.

4.4.2 Correctheid

Voor de toelatingsvoorwaarde tot de kandidatenverzameling bij General Match en de correctheid ervan beschouwen we de lemma's 4.7 en 4.8. Beide lemma's worden bewezen in [MWH02].

Lemma 4.7 ([MWH02]) *Als de sequentie S verdeeld wordt in J-sliding windows, dan is de eerste J-Sliding window die voorkomt in de deelsequentie $S[i:j]$ de $\left(\left\lceil\frac{i-1}{J}\right\rceil+1\right)$-de J-sliding window van S.*

Lemma 4.8 ([MWH02]) *Stel dat een datasequentie S_n opgedeeld wordt in J-Sliding windows van lengte ω en dat een querysequentie Q verdeeld wordt in J-Disjoint windows van dezelfde lengte. Als een deelsequentie $S[i:j]$ van lengte $|Q|$ een ϵ-match vormt met Q, dan vormt minstens één van de ingesloten windows $s_{a+n*\frac{\omega}{J}}^J$ $(0 \leq n \leq \rho-1)$ van $S[i:j]$ een $\epsilon/\sqrt{\rho}$-match met een J-Disjoint window $q_{(b+1,n+1)}^J$ van Q. Waarbij a gelijk is aan $\left\lceil\frac{i-1}{J}\right\rceil+1$, b gelijk is aan $(a-1)*J-i+1$ en ρ het aantal ingesloten windows is in $S[i:j]$, nl. $\left\lfloor\frac{|Q|-b}{\omega}\right\rfloor$. M.a.w.*

$$D(S[i:j],Q) \leq \epsilon \Rightarrow \bigvee_{n=0}^{\rho-1} D\left(s_{a+n*\frac{\omega}{J}}^J, q_{(b+1,n+1)}^J\right) \leq \epsilon/\sqrt{\rho}$$

Lemma 4.8 garandeert ons dat er zich zeker geen false dismissals zullen voordoen wanneer we werken met de volgende toelatingsvoorwaarde: een datasequentie S_n wordt opgenomen tot de verzameling als er minstens één J-Sliding window in S_n bestaat die matcht met een J-Disjoint window van de querysequentie Q.

4.4.3 Maximale windowgrootte

Om gebruik te kunnen maken van General Match, is het nodig dat de querysequentie minstens één window bevat. De maximale windowgrootte is dus wederom afhankelijk van de minimale lengte van de querysequentie. Lemma 4.9 geeft dit onderling verband weer. Het bewijs hiervan wordt gegeven in [MWH02].

Lemma 4.9 ([MWH02]) *Als de minimale lengte van de querysequentie gegeven wordt door Min(Q), dan is de maximale toegelaten windowgrootte gelijk aan* $\left\lfloor \frac{min(Q)-J+1}{J} \right\rfloor * J$.

Uit lemma 4.9 blijkt ook dat de waarde van J een belangrijke rol speelt bij de bepaling van de maximale windowgrootte. Hoe groter de waarde van J, hoe kleiner de maximale windowgrootte. Het is bij General Match dus mogelijk om voor een bepaald probleem een optimale maximale windowgrootte vast te leggen door een geschikte waarde voor J te kiezen.

4.5 Vergelijking

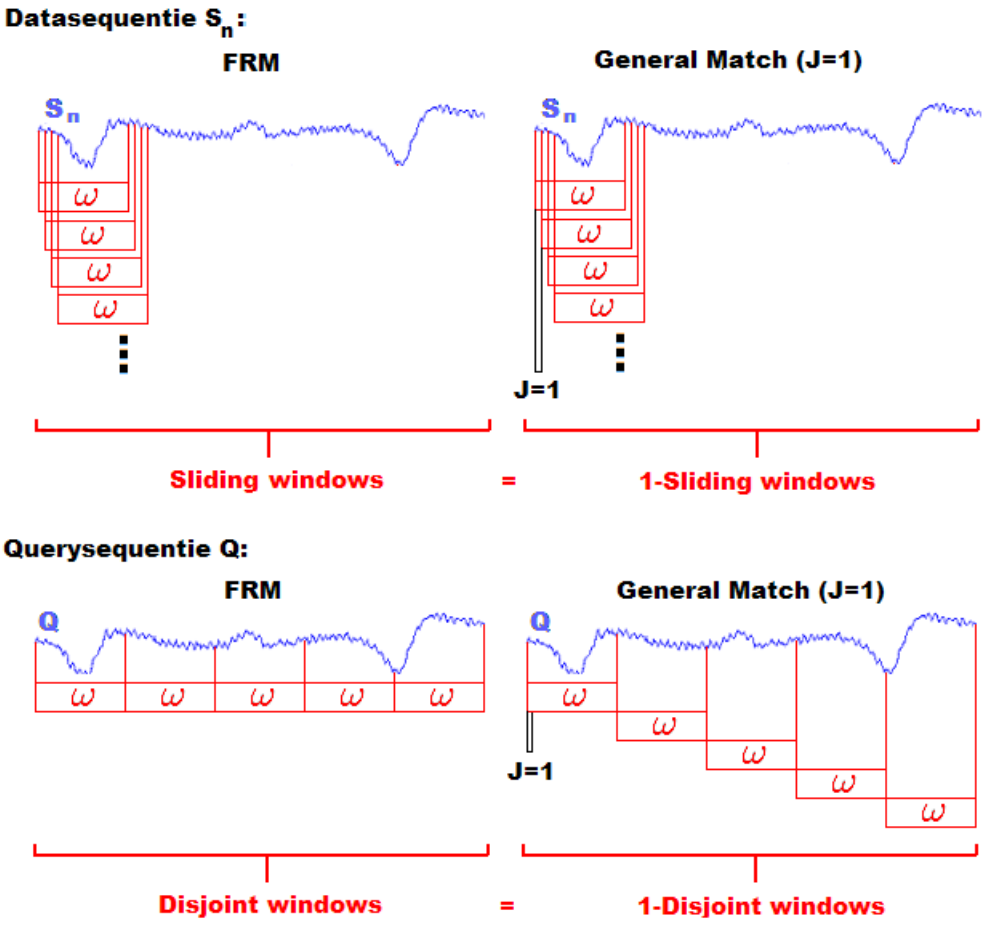
Tot slot van dit hoofdstuk overlopen we de voor- en nadelen van FRM, Dual Match en General Match. We hebben gezien dat, wegens het windows size effect, het wenselijk is om te kunnen werken met zo groot mogelijke windows. De maximale windowgrootte bij Dual Match is ongeveer half zo groot als deze van FRM. In geval van General Match is de maximale windowgrootte gelijk aan deze van FRM indien J de waarde één heeft. Des te groter de waarde van J, des te groter het verschil in maximale windowgrootte in het voordeel van FRM. Het vergroten van deze waarde heeft bij General Match dus een negatieve invloed op de maximale windowgrootte.

Een volgend vergelijkingspunt dat we beschouwen, is het aantal windows dat we bekomen door de datasequenties op te splitsen in windows van lengte ω. Met betrekking tot search performance en data-opslag is het wenselijk dit aantal zo klein mogelijk te houden. Dual Match haalt hierbij de beste resultaten omdat het dergelijke sequenties in disjoint windows opdeelt. FRM scoort op dit onderdeel het slechtst omdat het de datasequenties verdeeld in sliding windows. Bij General Match varieert het aantal windows, afkomstig van de datasequenties, afhankelijk van de gekozen J. Hoe groter de waarde van J, hoe kleiner het aantal windows.

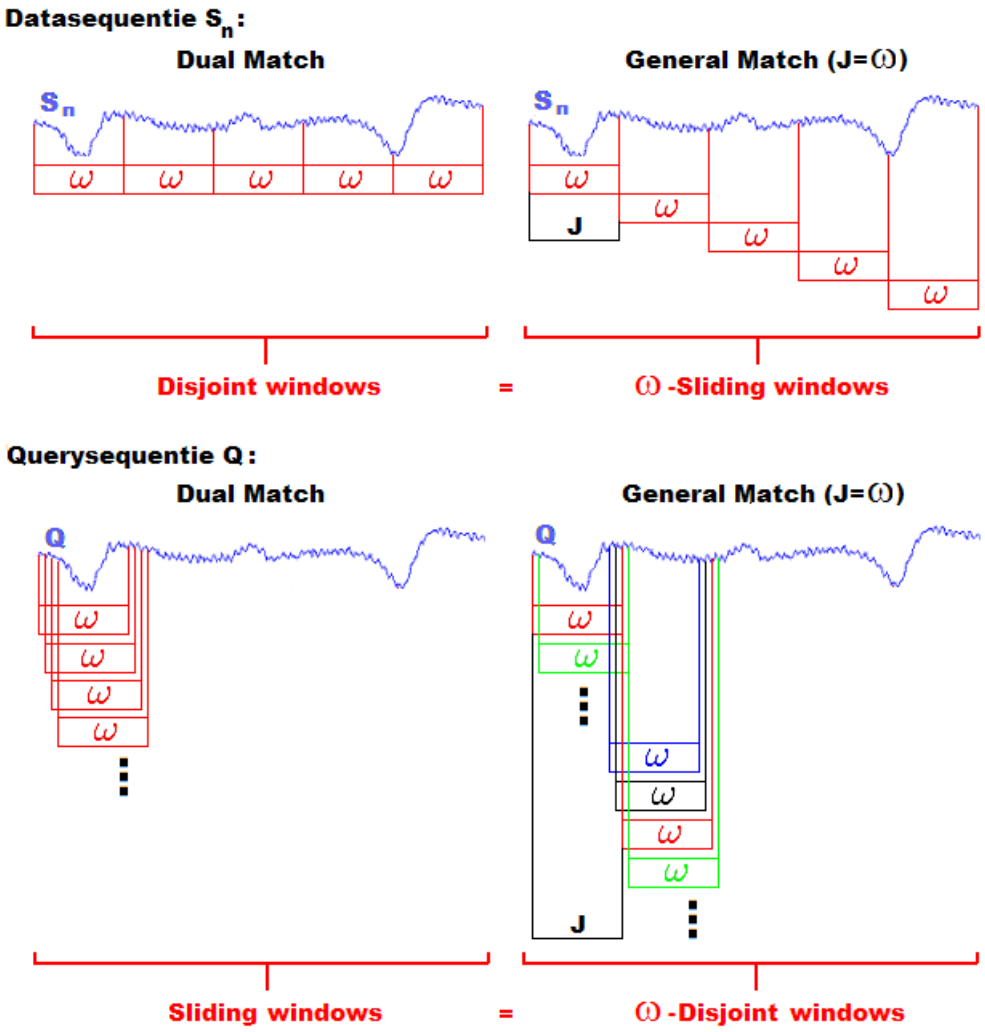
Zowel FRM als Dual Match kunnen als een speciaal geval van General Match beschouwd worden. FRM is het specifieke geval van General Match waarbij de waarde van J gelijk is aan één. Door de waarde van J gelijk te

stellen aan de lengte van de windows (ω) komt men terecht in de situatie van Dual Match. Beide situaties worden respectievelijk in figuren 4.8 en 4.9 weergegeven.

Door een geschikte waarde voor J te bepalen voor een bepaalde probleemsituatie, is het dus mogelijk om m.b.v. General Match gedeeltelijk te profiteren van zowel de voordelen van FRM als deze van Dual Match. De waarde van J zal dan ook meestal bepaald worden door het belang van de maximale windowgrootte af te wegen t.o.v. het belang van het aantal J-Sliding windows. Voor het verdere verloop van dit werk zullen we steeds General Match aanwenden als sliding window techniek. We zullen hierbij dan ook steeds proberen te zoeken naar een optimale waarde voor J.



Figuur 4.8: FRM als speciaal geval van General Match



Figuur 4.9: Dual Match als speciaal geval van General Match

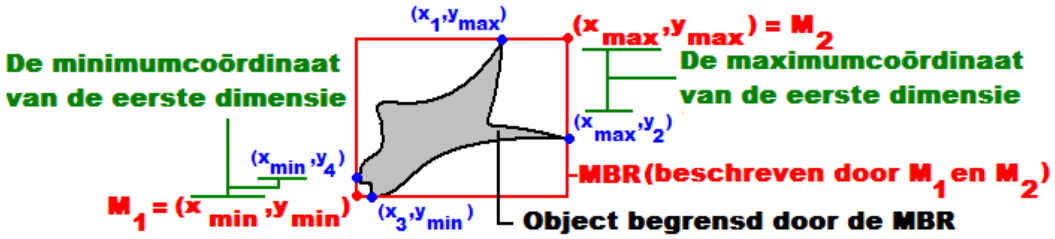
Hoofdstuk 5

Spatial Access Method

Met betrekking tot spatial databases kunnen we een tijdreeks beschouwen als een punt in een n-dimensionale ruimte. Een *Spatial Access Method (SAM)* wordt volgens [Nat] omschreven als een datastructuur voor het zoeken naar multidimensionale punten, lijnen, polygonen en andere geometrische lichamen. Tijdreeksen kunnen dus met behulp van een SAM geïndexeerd worden, wat de search performance gevoelig verbetert. In dit hoofdstuk bespreken we kort een veel gebruikte SAM op vlak van similarity search, namelijk de R-Tree.

5.1 Minimum Bounding Rectangle

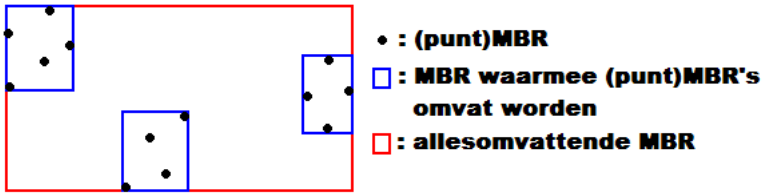
Een *minimum bounding rectangle (MBR)* is een geometrisch lichaam waarmee we een groep van één of meer geometrische lichamen kunnen begrenzen. Indien we werken in een n-dimensionale ruimte wordt een MBR beschreven door de twee punten $M_1 = (min_1, \ldots, min_n)$ en $M_2 = (max_1, \ldots, max_n)$. De coördinaten van deze punten worden gevonden door per dimensie i $(1 \leq i \leq n)$ zowel de minimum- $(= min_i)$ als de maximumcoördinaat $(= max_i)$ te zoeken tussen de coördinaten van alle objecten, die door de MBR in kwestie begrensd worden. Dit wordt verduidelijkt aan de hand van figuur 5.1.



Figuur 5.1: Minimum Bounding Rectangle

In dit werk zullen we, met betrekking tot een SAM, steeds werken met punten in een n-dimensionale ruimte. We zullen in dit hoofdstuk dan ook andere soorten geometrische lichamen (zoals lijnstukken,vierkanten,...) buiten beschouwing laten. Voor n-dimensionale punten wordt dezelfde methode gehanteerd als deze geïllustreerd in figuur 5.1. Een MBR, waarmee slechts één enkel n-dimensionaal punt begrensd wordt, is echter wel een speciaal geval. Zij nu het punt p een soortgelijk punt met coördinaten $(p_1, \ldots, p_n)$, dan wordt de MBR hiervan beschreven door M_1=$(p_1, \ldots, p_n)$ en M_2=$(p_1, \ldots, p_n)$. We zien dus duidelijk dat een MBR die slechts één enkel punt begrensd niets meer voorstelt dan dit punt zelf (M_1=M_2).

Ten slotte vermelden we dat het tevens mogelijk is om met behulp van een MBR andere MBR's te begrenzen. De coördinaten van een dergelijk MBR worden nog steeds volgens dezelfde methode berekend. Het enige verschil is echter dat er nu per dimensie naar de minimum- en de maximumcoördinaat gezocht wordt van al de MBR's die door de MBR in kwestie begrensd worden. Dergelijke MBR wordt geïllustreerd in figuur 5.2.



Figuur 5.2: Een MBR waarmee andere MBR's begrensd worden.

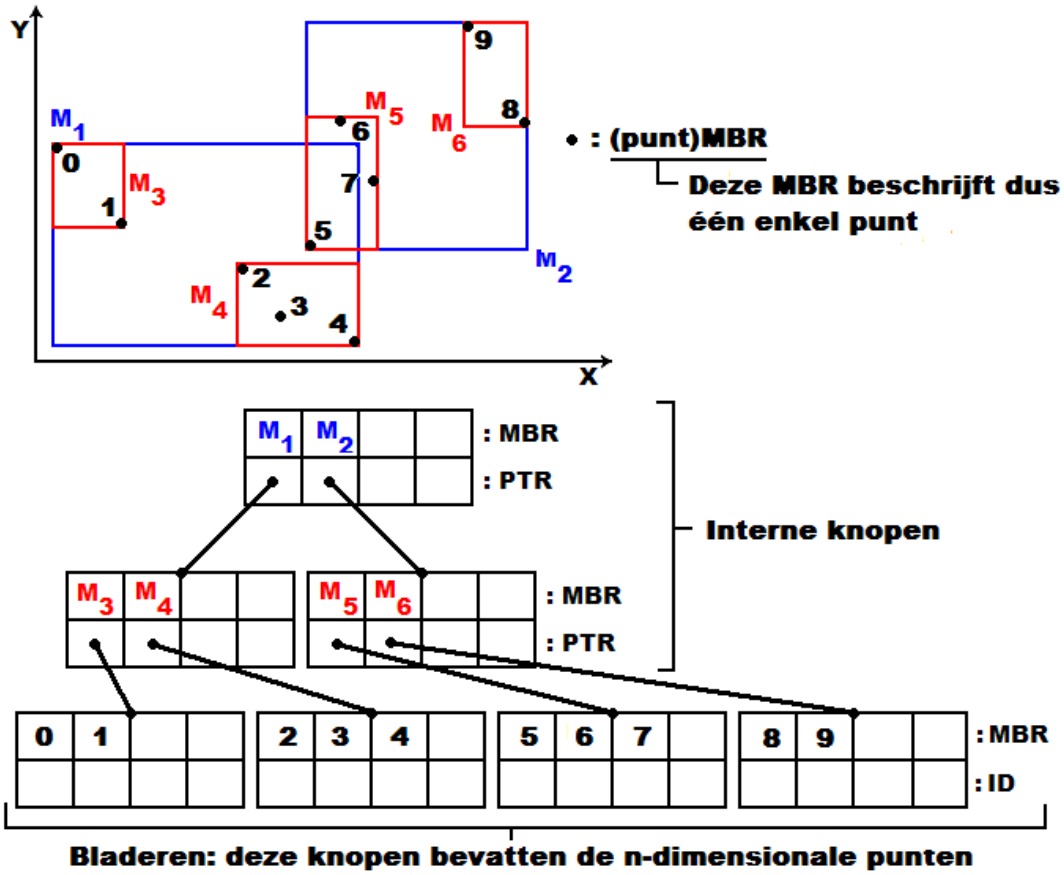
5.2 R-Tree

De R-Tree werd voor het eerst voorgesteld in [Gut84] en werd speciaal ontworpen voor het indexeren van multidimensionale objecten. In ons geval richten we ons specifiek op het indexeren van multidimensionale punten. We zullen nu de opbouw en de eigenschappen van deze boom behandelen.

5.2.1 Structuur

Om te beginnen wordt elk multidimensionaal punt in de R-Tree voorgesteld d.m.v. zijn MBR. De knopen van een R-Tree kunnen in twee categorieën onderverdeeld worden, namelijk interne knopen (internal nodes) en bladeren (leaf nodes). Elk blad kan tuples van de vorm (id,mbr) bevatten. In het id-veld wordt een sleutelwaarde bewaard, waarmee het overeenkomstige multi-

dimensionaal punt in de spatial database kan gevonden worden. De MBR die dit punt omvat, wordt bijgehouden in het *mbr*-veld. Een interne knoop daarentegen bevat tuples van de vorm (*ptr*,*mbr*). M.b.v. het *ptr*-veld wordt een pointer bijgehouden naar de onderhangende knoop (child node) van het huidige tuple. De MBR, die alle MBR's van de onderhangende knoop omvat, wordt bewaard m.b.v. het *mbr*-veld. Figuur 5.3 verduidelijkt de besproken structuur.



Figuur 5.3: R-tree: voorbeeld [Gut84]

Als M het maximaal aantal tuples per knoop (interne knoop of blad) is en indien $m \leq \frac{M}{2}$ het minimum aantal tuples voorstelt, dan moet een R-Tree volgens [Gut84] aan de volgende bijkomende voorwaarden voldoen:

- De top (root) van de boom heeft minstens twee onderhangende knopen, behalve in het geval dat deze top een blad is.
- Elk knoop (zowel blad als interne knoop) bevat tussen de m en M aantal tuples, behalve als deze knoop de top van de boom is.

- Alle bladeren bevinden zich op hetzelfde niveau.

De diepte van de R-Tree is hoogstens gelijk aan $\lceil log_m N \rceil - 1$, waarbij N het aantal te indexeren punten voorstelt. Dit komt door het feit dat er zich per knoop minstens m aantal tuples bevinden.

Bij de R-Tree wordt er ook steeds voor gezorgd dat de boom over een evenwichtig gestructureerde vorm beschikt. De voorgaande voorwaarden dragen hier al een klein beetje aan bij. Hoe de R-Tree zorgt voor een evenwichtige structuur zullen we echter niet behandelen in dit werkstuk. We bekijken daarentegen wel hoe we een R-Tree kunnen gebruiken bij similarity search. Hier zullen we in latere hoofdstukken nog op terugkomen.

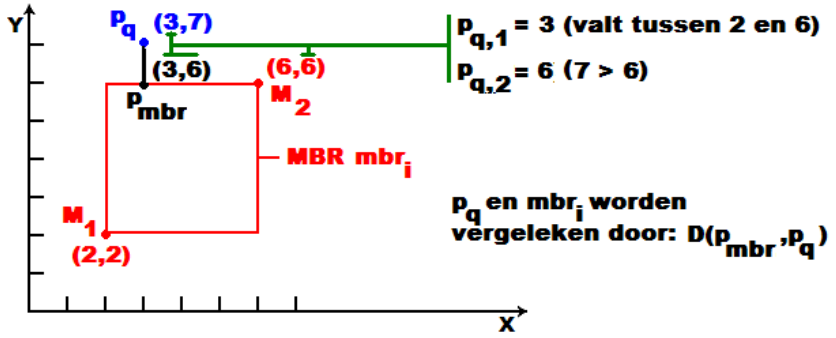
5.2.2 Range search

Ten slotte bekijken we hoe we alle n-dimensionale punten in de R-Tree kunnen terugvinden die binnen een bepaald bereik ϵ van een ander n-dimensionaal punt p_q liggen. De coördinaten van punt p_q worden gegeven door $(p_{q,1}, \ldots, p_{q,n})$. Om de te zoeken punten te vinden, is het dus nodig dat p_q vergeleken wordt met de MBR's in de R-tree. De vraag is nu hoe we dit punt p_q kunnen vergelijken met deze MBR's. Hierbij doen we beroep op een afstandsfunctie $D()$. Welke afstandsfunctie hier juist voor gebruikt wordt, is vaak afhankelijk van de situatie waarin de R-Tree gebruikt wordt. Hier komen we in latere hoofdstukken nog op terug.

Het is nodig een onderscheid te maken tussen de MBR's in de R-Tree. In de R-Tree bevinden zich namelijk twee soorten van MBR's. De MBR's in de bladeren van de R-Tree representeren de n-dimensionale punten in de R-Tree. Zoals eerder vermeld stellen de MBR's in de bladeren niets meer voor dan het n-dimensionale punt p dat ze representeren. Om de afstand tussen een dergelijk MBR, dewelke het punt p representeert, en een punt p_q te berekenen, berekenen we simpelweg de afstand tussen p en p_q, nl. $D(p,p_q)$. De MBR's in de interne knopen daarentegen zijn MBR's die de onderliggende MBR's omvatten. Een probleem dat zich voordoet bij het berekenen van de afstand tussen een punt en dergelijke MBR's, is dat een punt voor iedere dimensie i beschreven wordt door één enkele waarde, nl. $p_{q,i}$. De MBR's in de interne knopen daarentegen worden voor iedere dimensie i beschreven door twee waarden, nl. min_i en max_i. Om dit op te lossen gaan we de MBR in kwestie tijdelijk beschrijven aan de hand van één n-dimensionaal punt p_{mbr} i.p.v. twee punten (M_1 en M_2). Dit punt p_{mbr} wordt beschreven door de coördinaten $(p_{mbr,1}, \ldots, p_{mbr,n})$, waarbij

$$p_{mbr,i} = \begin{cases} p_{q,i} & \text{als } min_i \leq p_{q,i} \leq max_i \\ max_i & \text{als } p_{q,i} > max_i \\ min_i & \text{als } p_{q,i} < min_i \end{cases}$$

Het punt p_q en een MBR mbr_i liggen m.a.w. binnen een bereik ϵ als D(p_{mbr},p_q) $\leq \epsilon$. Door de opbouw van het punt p_{mbr} kunnen we D(p_{mbr},p_q) opvatten als de kortste afstand tussen het punt p_q en de MBR mbr_i. Deze situatie wordt verduidelijkt m.b.v. figuur 5.4. Hier zien we hoe een MBR mbr_i vergeleken wordt met een punt p_q in de twee-dimensionale ruimte.



Figuur 5.4: Het vergelijken van een punt en een MBR

Bij het zoeken naar alle n-dimensionale punten in de R-Tree die binnen een bepaald bereik ϵ van een ander n-dimensionaal punt p_q liggen, beginnen we steevast vanuit de top van de boom. Van daaruit wordt naar beneden toe gewerkt, richting het niveau van de bladeren. In pseudocode 1 wordt een zoekfunctie beschreven om dit probleem aan te pakken. Vermits we steeds in de top van de boom beginnen, luidt de aanroep van deze functie dus als volgt $ZoekPunten(p_q,top,\epsilon)$.

Met behulp van het zoekalgoritme uit pseudocode 1 wordt er dus doorheen de boom gelopen. Hierbij wordt p_q steeds vergeleken met al de MBR's in de huidige knoop. Indien $D(p_{mbr},p_q) > \epsilon$ kan de tak, die hangt onder de MBR (overeenkomstig met p_{mbr}), volledig weggesnoeid worden. Figuur 5.5 toont ons de situatie waarbij takken van een R-Tree kunnen weggesneden worden. Een weggesnoeide tak zou immers toch geen bijdrage meer kunnen leveren aan het eindresultaat. Dit komt omdat als $D(p_{mbr},p_q) > \epsilon$, dat dan voor elk punt p_i, bevat in de MBR overeenkomstig met p_{mbr}, geldt dat $D(p_i,p_q) > \epsilon$. We verduidelijken dit m.b.v. figuur 5.6. Door het feit dat er meestal enkele takken weggesnoeid kunnen worden, is het vaak niet nodig dat de boom steeds volledig moet doorlopen worden. Dit gegeven levert uiteraard een positieve bijdrage aan de efficiëntie van het zoeken.

Pseudocode 1 ZoekPunten(PUNT p_q,KNOOP *huidigeKnoop*,BEREIK ϵ)

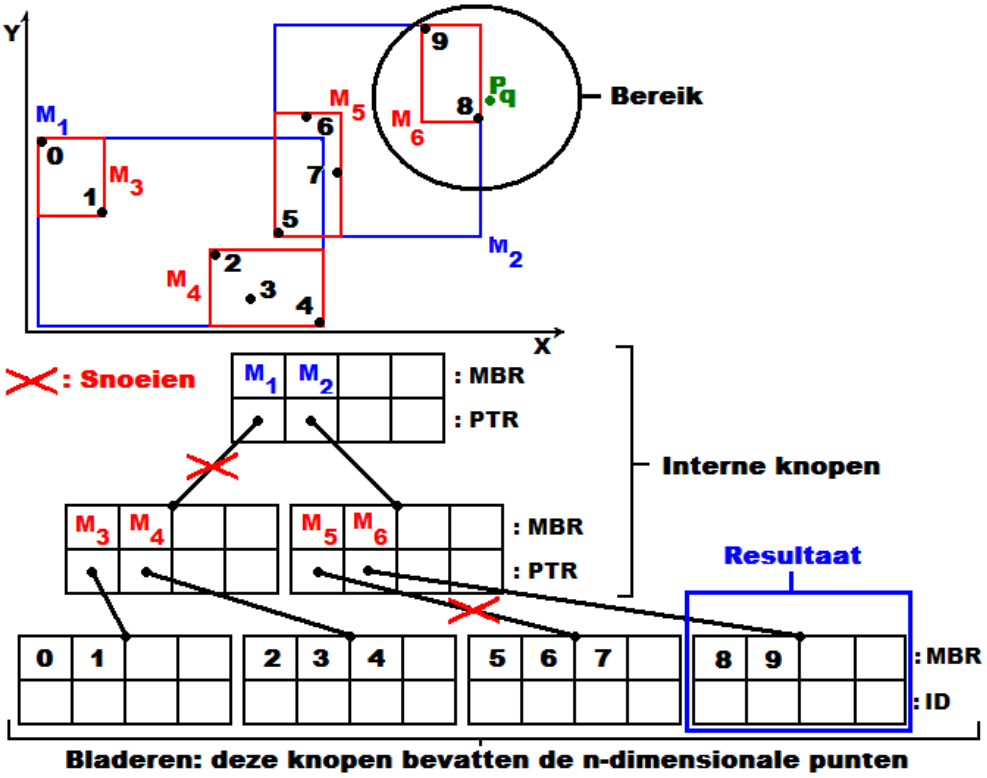
```
if (huidigeKnoop is een blad) then
  for (alle tuples (id_i,mbr_i) van huidigeKnoop, i = 1,...,#tuples) do
    p_mbr = berekenPunt(mbr_i,p_q); // zoals eerder beschreven
    if (D(p_mbr,p_q)≤ ε) then
      resultaat += id_i;
    end if
  end for
else if (huidigeKnoop is een interne knoop) then
  for (alle tuples (ptr_i,mbr_i) van huidigeKnoop, i = 1,...,#tuples) do
    p_mbr = berekenPunt(mbr_i,p_q); // zoals eerder beschreven
    if (D(p_mbr,p_q)≤ ε) then
      resultaat += ZoekPunten(p_q,GeefKnoop(ptr_i),ε);
    end if
  end for
end if
return resultaat;
```

5.2.3 Implementatie

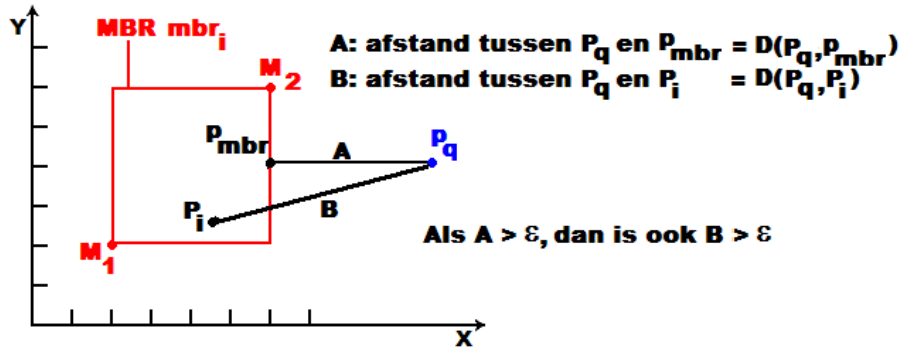
Voor de implementatie van de R-Tree doen we in dit werk beroep op [Had]. Deze implementatie is gebaseerd op [Gut84].

5.3 Andere SAM's

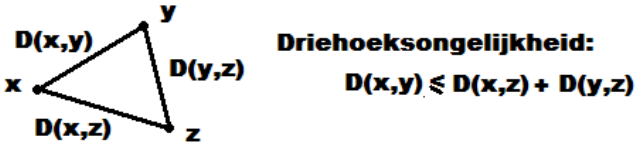
In sectie 5.2.2 hebben we gezien dat de R-Tree met betrekking tot range query's een beroep doet op een afstandsfunctie $D()$. In meerdere studies over similarity search, zoals bijvoorbeeld [YJF98] en [HS03], wordt erop gewezen dat er bij vele, doch niet alle, SAM's een beroep wordt gedaan op de *driehoeksongelijkheid* met betrekking tot o.a. dergelijke query's. Een afstandsfunctie $D()$ voldoet aan de driehoeksongelijkheid als $\forall x, y, z \in \mathbb{R}^n$ geldt dat $D(x,y) \leq D(x,z) + D(z,y)$ (zie figuur 5.7). Tevens hebben we gezien dat similarity search op verschillende soorten van afstandsmetingen kan gebaseerd zijn. De keuze van afstandsmeting zou dan wel eens beïnvloed kunnen worden door het feit dat deze afstandsmeting al dan niet aan de driehoeksongelijkheid voldoet. Vooral wanneer we hierbij een specifieke SAM wensen te gebruiken.



Figuur 5.5: R-tree: zoekalgoritme [Gut84]



Figuur 5.6: Werking van het zoekalgoritme.



Figuur 5.7: Driehoeksongelijkheid

Deel I

Euclidische metriek

Hoofdstuk 6

Euclidische afstand

In hoofdstuk 3 hebben we gezien dat er bij similarity search steevast beroep wordt gedaan op een afstandsfunctie. Tevens werd er een opdeling gemaakt afhankelijk van de te gebruiken afstandsfunctie. In dit deel behandelen we het specifieke geval van similarity search gebaseerd op de Euclidische metriek. Dit hoofdstuk verschaft ons de nodige kennis over dergelijke metriek.

6.1 Afstandsfunctie

De Euclidische afstand tussen twee punten kunnen we opvatten als de lengte van het lijnstuk, dat deze punten met elkaar verbindt. Dit kunnen we in termen van de n-dimensionale ruimte als volgt definiëren:

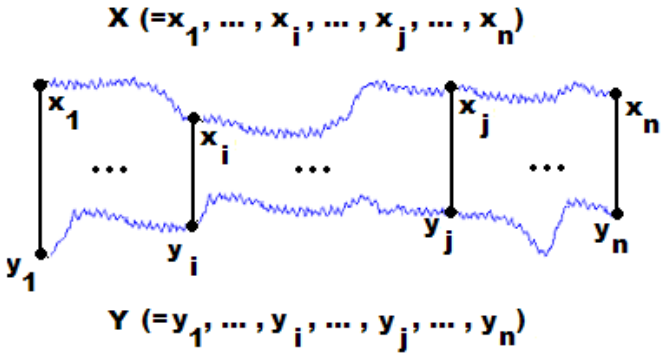
Definitie 6.1 *Zij* $x\ (= x_1, \ldots, x_n)$ *en* $y\ (= y_1, \ldots, y_n)$ *twee* n*-dimensionale punten, dan wordt de* Euclidische afstand *tussen beide punten gegeven door:*

$$D(x,y) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2}.$$

Definitie 6.2 legt vast wat we verstaan onder gelijke sequenties in het geval van similarity search gebaseerd op de Euclidische afstand. Hierbij zien we duidelijk de gelijkenis met het begrip ϵ-match, zoals beschreven werd in sectie 3.1. Figuur 6.1 dient ter illustratie van deze definitie.

Definitie 6.2 ([CFY03]) *Gegeven een tolerantie* ϵ*, dan zijn de twee sequenties* $X\ (= x_1, \ldots, x_n)$ *en* $Y\ (= y_1, \ldots, y_n)$ *van gelijke lengte* n gelijk *als*

$$D(X,Y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \leq \epsilon.$$



Figuur 6.1: Similarity search op basis van de Euclidische metriek.

Met betrekking tot de Euclidische afstand gelden de volgende interessante eigenschappen:

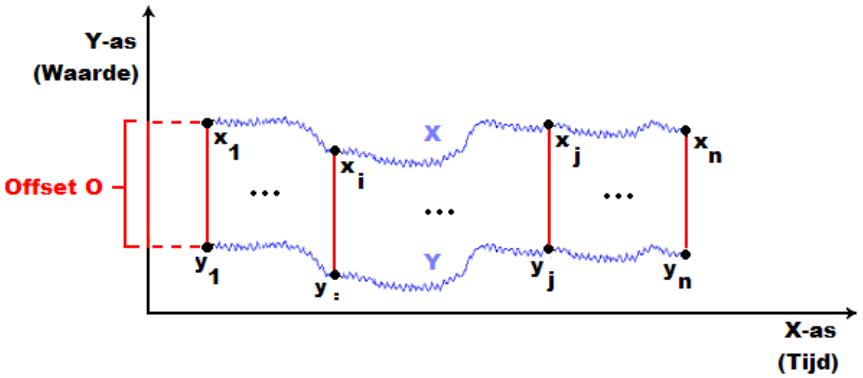
- $D(X,X) = 0$
- $D(X,Y) = 0 \Leftrightarrow X = Y$
- $D(X,Y) = D(Y,X)$ (Symmetrisch)
- $D(X,Y) \leq D(X,Z) + D(Z,Y)$ (Driehoeksongelijkheid)

Zoals we gemerkt hebben in sectie 5.3, doen heel wat SAM's beroep op deze laatste eigenschap. Vandaar dat de Euclidische afstandsmeting o.a. interessant blijkt te zijn met betrekking tot het gebruik van SAM's.

6.2 Verticale verschuiving

Een probleem dat mogelijk kan optreden bij het vergelijken van sequenties, is dat van *verticale verschuivingen* (*v-shifts*). Hiervan is sprake indien twee sequenties een haast identiek verloop over de tijd hebben. Bijkomend ligt iedere waarde van de ene sequentie steeds ongeveer een constante waarde (offset) hoger dan de overeenkomstige waarde in de tijd van de andere sequentie. We verduidelijken deze situatie m.b.v. figuur 6.2. Op deze figuur zien we dat sequentie X hier steeds een offset O hoger ligt dan sequentie Y. Qua vormgeving zijn beide sequenties zo goed als identiek. Hieruit blijkt dus dat het probleem van verticale verschuivingen opgevat kan worden als een verschuiving over de Y-as.

De Euclidische afstandsmeting, zoals in definitie 6.2, is niet bestand tegen dit probleem van verticale verschuivingen. Definitie 6.3 biedt hier echter



Figuur 6.2: Het probleem van verticale verschuivingen

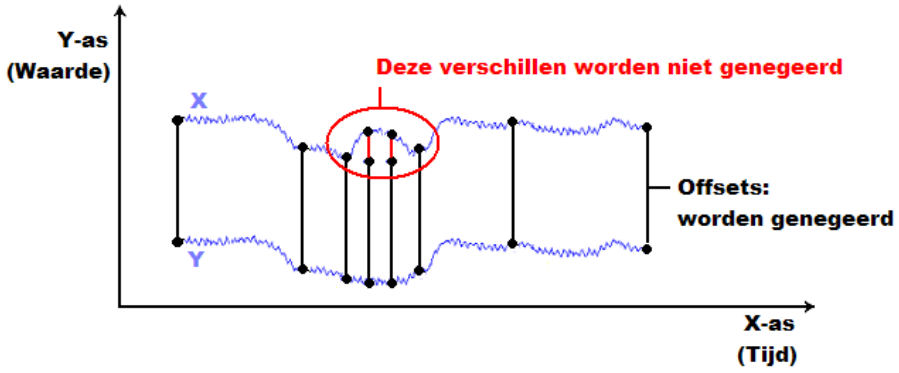
een oplossing, door een kleine aanpassing aan deze afstandsmeting aan te brengen. Bij deze definitie worden, in vergelijking met definitie 6.2, alle verticale offsets langsheen de gehele x-as genegeerd. Deze nieuwe definitie wordt verduidelijkt met behulp van figuur 6.3.

Definitie 6.3 ([CFY03]) *Gegeven een tolerantie ϵ, dan zijn de twee sequenties X en Y van gelijke lengte n* v-shift gelijk *als*

$$D(X,Y) = \sqrt{\sum_{i=1}^{n} ((x_i - y_i) - (x_{avg} - y_{avg}))^2},$$

waarbij

$$x_{avg} = \frac{1}{n}\sum_{i=1}^{n} x_i \ \ en \ \ y_{avg} = \frac{1}{n}\sum_{i=1}^{n} y_i.$$



Figuur 6.3: Aangepaste Euclidische afstandsmeting.

Voor het verdere vervolg gaan we echter niet dieper in op het probleem van verticale verschuivingen. Om dergelijke problemen quasi uit te sluiten, stellen we voor om de ECG-sequenties, indien nodig, te normaliseren in de richting van de Y-as. Deze operatie kan plaatsvinden bij het initialiseren van de database of bij het invoegen van nieuwe ECG-sequenties. Het enige extra werk dat dit op het vlak van similarity search met zich mee zou kunnen brengen, is dat de querysequentie mogelijk eerst genormaliseerd zal moeten worden.

Hoofdstuk 7

GEMINI

In dit hoofdstuk behandelen we een veelgebruikte algemene methode voor similarity search gebaseerd op de Euclidische metriek. Deze methode is bekend onder de naam GEMINI (GEneric Multimedia INdexIng).

7.1 Algemeen

De *GEMINI-methode* werd voorgesteld in [FRM94] en [Fal96] met als doel similarity search van allerhande multimedia-objecten. Voorbeelden van dergelijke objecten zijn afbeeldingen, tijdreeksen, muziekopnamen, ... We richten ons hier natuurlijk volledig op het aspect van tijdreeksen of sequenties. De GEMINI-methode werd voornamelijk ontwikkeld om het probleem van whole matching aan te pakken. Hoe we deze methode uitbreiden met het oog op subsequence matching, wordt behandeld in sectie 7.2.

Typerend aan de GEMINI-methode is dat er beroep wordt gedaan op een SAM om de datasequenties te indexeren. Omdat de efficiëntie van een SAM te fel afneemt bij een te hoge dimensionaliteit van de sequenties, dient deze hoge dimensionaliteit gereduceerd te worden. We bespreken nu het geval waarbij we de R-Tree als SAM gebruiken. Alvorens een n-dimensionale datasequentie S_i aan de R-Tree wordt toegevoegd, moet deze sequentie afgebeeld worden op een punt p_i in de m-dimensionale ruimte. Hierbij geldt dat $n > m$. De datasequenties worden m.a.w. als m-dimensionale punten voorgesteld in de R-Tree. Technieken voor het reduceren van de dimensionaliteit van tijdreeksen zullen in hoofdstuk 8 aan bod komen.

De algemene werking van de GEMINI-methode voor similarity search van tijdreeksen, wordt beschreven aan de hand van de volgende stappen:

- **Stap 1:** De n-dimensionale querysequentie Q wordt op een punt p_q in de m-dimensionale ruimte afgebeeld m.b.v. een reductietechniek $\mathcal{F}$.

Dit kunnen we ook uitdrukken als $\mathcal{F}(Q)$=p_q.

- **Stap 2:** Door gebruik te maken van de R-Tree worden alle punten p_i teruggegeven waarvoor geldt: $D_{reductie}(p_i,p_q) \leq \epsilon$. De afstandsmeting $D_{reductie}()$ is afhankelijk van de gebruikte reductietechniek. Vaak is $D_{reductie}()$ echter gewoon de Euclidische afstandsmeting. We zullen hier, bij het hoofdstuk over reductietechnieken (hoofdstuk 8), nog op terugkomen. Er zal daar per reductietechniek aangegeven worden welke afstandsfunctie $D_{reductie}()$ er juist gebruikt wordt. De voorwaarde $D_{reductie}(p_i,p_q) \leq \epsilon$ betekent hetzelfde als: $D_{reductie}\left(\mathcal{F}\left(S_i\right),\mathcal{F}\left(Q\right)\right) \leq \epsilon$. Hoe we precies alle punten in de R-Tree vinden die aan deze voorwaarde voldoen, werd reeds besproken in sectie 5.2.2. Uiteraard werken we daar dan met $D_{reductie}()$ als afstandsmeting.

- **Stap 3:** Voor elk gevonden punt p_i grijpen we terug naar de overeenkomstige originele datasequentie S_i. Vervolgens elimineren we alle false alarms door alleen die datasequenties S_i te aanvaarden waarvoor geldt: $D(S_i,Q) \leq \epsilon$.

De tweede stap van de GEMINI-methode kan opgevat worden als een 'quick-and-dirty'-test. Met behulp van deze test kan een groot aantal kandidaatsequenties genegeerd worden. Dit kan zelfs zonder dat er zich false dismissals zullen voordoen, wel in de veronderstelling dat de reductietechniek $\mathcal{F}$ voldoet aan lemma 7.1. Met betrekking tot dit lemma en de voorgaande stappen stelt $D()$ in ons geval de Euclidische afstandsmeting voor. Het bewijs van dit lemma is terug te vinden in [FRM94]. In tegenstelling tot false dismissals is het echter wel mogelijk dat deze test false alarms doorlaat. Deze worden er dan in de derde stap van de GEMINI-methode uitgehaald.

Lemma 7.1 ([FRM94]) *Om te kunnen garanderen dat er zich geen false dismissals voordoen bij de whole matching query's, is het nodig dat de reductietechniek $\mathcal{F}$ voldoet aan de volgende voorwaarde:*

$$D_{reductie}(\mathcal{F}(O_1),\mathcal{F}(O_2)) \leq D(O_1,O_2)$$

voor elk paar van objecten O_1,O_2.

7.2 Subsequence matching

De besproken GEMINI-methode biedt ons echter nog geen oplossing om het subsequence matching probleem aan te pakken. In sectie 4.1 hebben we evenwel gezien hoe het mogelijk is deze methode daartoe uit te breiden door

gebruik te maken van een sliding window techniek. De volgende stappen beschrijven de resulterende subsequence matching methode:

- **Initialisatiestap:** Elke datasequentie S_i wordt m.b.v. een sliding window techniek opgesplitst in windows van lengte ω. Vervolgens past men de reductietechniek $\mathcal{F}$ toe op elk van de bekomen windows. Elke n-dimensionale window wordt m.a.w. gereduceerd naar een m-dimensionaal punt. Ieder punt stelt dus een klein stuk (één window) voor van de desbetreffende datasequentie S_i in de m-dimensionale ruimte. Al deze punten worden vervolgens aan een SAM toegevoegd. Deze stap wordt uiteraard enkel uitgevoerd bij het initialiseren van de database of bij het invoegen van nieuwe datasequenties. In dit laatste geval dienen de reeds eerder ingevoegde datasequenties niet opnieuw ingevoegd te worden.

- **Stap 1:** De querysequentie Q wordt in windows van lengte ω opgesplitst. Ook hier wordt iedere window omgezet naar een m-dimensionaal punt m.b.v. de reductietechniek $\mathcal{F}$. Elk punt stelt dus een klein stuk (één window) voor van de querysequentie. We zullen deze punten voor de eenvoud benoemen als de querypunten.

- **Stap 2:** Op analoge wijze als in stap twee van de GEMINI-methode worden hier alle querypunten vergeleken met al de punten in de SAM. De resulterende verzameling van punten vormt de kandidatenverzameling, zoals beschreven in hoofdstuk 4. De overeenkomstige window van elk punt in deze verzameling matcht dus met één van de windows van de querysequentie.

- **Stap 3:** Voor ieder punt in de kandidatenverzameling grijpen we terug naar zijn overeenkomstige datasequentie S_i. Door voor elk punt bij te houden met welk querypunt het juist overeenkomt, kunnen we een bepaalde regio S_{regio} van lengte $|Q|$ in de overeenkomstige datasequentie afbakenen. Hierna wordt de gevonden regiosequentie m.b.v. de afstandsfunctie vergeleken met de querysequentie, $D(Q,S_{regio}) \leq \epsilon$. Op deze manier worden de false alarms uit de kandidatenverzameling verwijderd en blijven alleen de (subsequence) matchings over.

In voorgaande stappen herkennen we een duidelijke overeenkomst met het meerstappenproces uit sectie 4.1. Bij dat proces werd er, zonder er dieper op in te gaan, een beroep gedaan op een (whole) matching-algoritme. De zojuist behandelde stappen zijn het specifieke geval van dit meerstappenproces waarbij de GEMINI-methode gebruikt wordt als matching-algoritme.

Hoofdstuk 8

Reduceren van de dimensionaliteit

Doorheen de jaren is er veel onderzoek verricht naar het reduceren van de dimensionaliteit van tijdreeksen. We zullen nu enkele technieken daartoe van naderbij beschouwen.

8.1 Discrete Fourier Transformatie

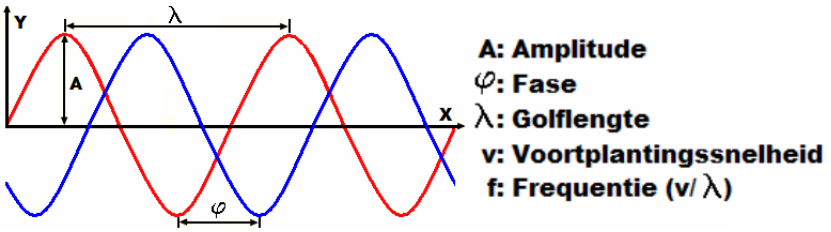
De *discrete Fourier transformatie*, of afgekort *DFT*, werd in [AFS93] voorgesteld als één van de eerste technieken voor het reduceren van de dimensionaliteit van tijdreeksen. Vooraleer we ons hier verder in verdiepen, zullen we eerst enkele inleidende basisbegrippen overlopen.

8.1.1 Basisbegrippen

De frequentie en amplitude zijn typerende begrippen met betrekking tot golven. Beide begrippen worden uitgelegd aan de hand van *sinus- en cosinusgolven*. Dergelijke golven zijn gekenmerkt door hun sinusoïdaal verloop over de tijd. Een sinusgolf wordt beschreven door $A\,sin\,(2\pi ft - \varphi)$, waarbij:

- t = de tijd
- A = de amplitude
- f = de frequentie
- φ = de fase

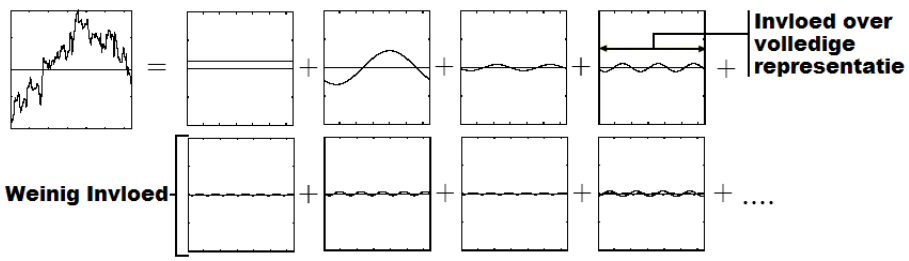
De grootte of de sterkte van een sinusgolf wordt weergegeven door de *amplitude* A. De *frequentie* f van een golf geeft de verhouding weer tussen de golflengte λ en de snelheid v waarmee de golf zich voortplant ($f = \frac{v}{\lambda}$). Dit begrip kan dus opgevat worden als het aantal keren dat een bepaalde gebeurtenis voorvalt per tijdseenheid. De *fase* geeft de verschuiving van een golf weer t.o.v. de X-as. De situatie voor cosinusgolven is geheel analoog. Het verschil tussen deze golven is dat er zich tussen beide golven een faseverschil van $\pi/2$ bevindt. Figuur 8.1 dient ter verduidelijking van de zojuist besproken begrippen.



Figuur 8.1: Frequentie en amplitude van een golf.

8.1.2 Theorie

Het basisidee achter DFT is dat eender welke tijdreeks beschreven kan worden door een samenstelling van een eindig aantal sinus- en/of cosinusgolven. Dit idee wordt geïllustreerd in figuur 8.2. Tevens kunnen we bij deze figuur opmerken dat sommige van deze golven slechts een kleine rol spelen in de samenstelling van de tijdreeks. Dit komt omdat deze golven slechts over een zeer kleine amplitude beschikken. Dergelijke golven zouden in principe verwaarloosd kunnen worden zonder dat er hierbij veel informatie over de originele tijdreeks verloren gaat. We zien echter wel dat elke golf een bijdrage levert over de hele lengte van de representatie van een tijdreeks.



Figuur 8.2: Een tijdreeks samengesteld uit sinus/cosinusgolven. [WAA00]

Een tijdreeks of sequentie S wordt, door de DFT erop toe te passen, omgezet naar een verzameling van $n(=|S|)$ aantal sinus/cosinusgolven. Elke golf zal hierbij voorgesteld worden door zijn complexe representatie. Naar deze complexe waarden wordt verwezen als de *Fourier coëfficiënten*. De DFT van een sequentie $S\,(= S_0, \ldots S_{n-1})$ wordt gegeven door

$$s_f = \frac{1}{\sqrt{n}} \sum_{i=0}^{n-1} S_i exp\left(\frac{-j2\pi if}{n}\right)$$

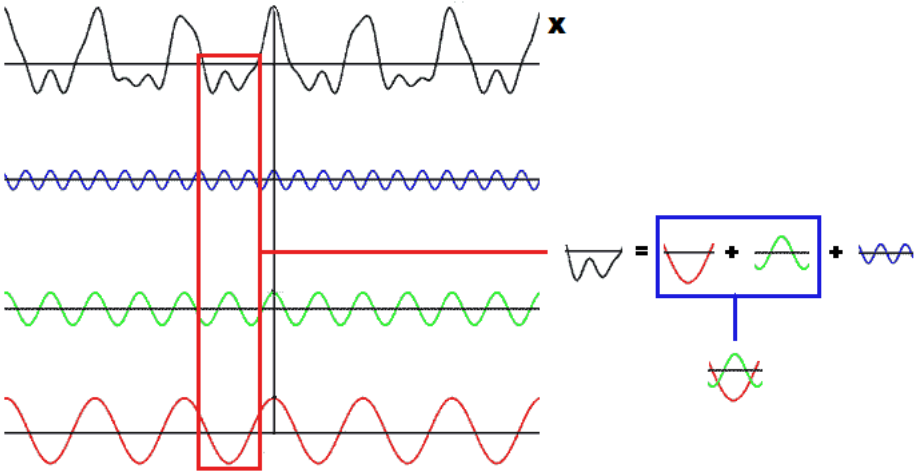
waarbij $f = 0, 1, \ldots, n-1$, $j = \sqrt{-1}$ en $exp\,(j\phi) \equiv cos\,(\phi) + j\,sin\,(\phi)$. Het is duidelijk dat elke Fourier coëfficiënt s_f, op uitzondering van s_0, een complex getal voorstelt.

Via de inverse DFT is het tevens mogelijk de originele sequentie S te bekomen, vertrekkende vanuit de Fourier coëfficiënten $s_0, \ldots s_{n-1}$. Deze inverse transformatie wordt beschreven door

$$S_i = \frac{1}{\sqrt{n}} \sum_{f=0}^{n-1} s_f exp\left(\frac{j2\pi fi}{n}\right)$$

waarbij $i = 0, 1, \ldots, n-1$, $j = \sqrt{-1}$ en $exp\,(j\phi) \equiv cos\,(\phi) + jsin\,(\phi)$.

Het toepassen van de DFT op een sequentie S resulteert dus in $n(=|S|)$ Fourier coëfficiënten. De dimensionaliteit van S werd m.a.w. dus nog steeds niet gereduceerd. Dit kunnen we echter verwezenlijken door enkel en alleen de eerste $k(< n)$ coëfficiënten te gebruiken om de sequentie voor te stellen. De overige $(n\text{-}k)$ coëfficiënten worden dus volledig genegeerd. Hierdoor werken we uiteraard slechts verder met een benadering van de originele sequentie. In figuur 8.3 zien we hoe een sequentie X kan benaderd worden door gebruik te maken van de eerste drie Fourier coëfficiënten.



Figuur 8.3: Benadering van een sequentie X m.b.v. de DFT.

In het vorige hoofdstuk hebben we vastgesteld dat enkel indien de gebruikte reductietechniek voldoet aan lemma 7.1, we kunnen garanderen dat er zich bij de GEMINI-methode geen false dismissals zullen voordoen. De eerste stap tot deze garantie wordt gezet d.m.v. stelling 8.1.

Stelling 8.1 ([Fal96]) (Stelling van Parseval) *Zij $s_0, \ldots, s_{n-1}$ de Fourier coëfficiënten van een sequentie $S(=S_0, \ldots, S_{n-1})$, dan geldt*

$$\sum_{i=0}^{n-1} |S_i|^2 = \sum_{f=0}^{n-1} |s_f|^2$$

Door gebruik te maken van deze stelling kan vrij eenvoudig afgeleid worden dat de Euclidische afstand bewaard blijft onder het toepassen van de DFT. Voor de Euclidische afstandsmeting tussen twee sequenties $S(=S_0, \ldots, S_{n-1})$ en $Q(=Q_0, \ldots, Q_{n-1})$ geldt namelijk

$$\begin{aligned} D(S,Q) &= \left(\sum_{i=0}^{n-1} |S_i - Q_i|^2\right)^{\frac{1}{2}} =^1 \left(\sum_{f=0}^{n-1} |s_f - q_f|^2\right)^{\frac{1}{2}} \\ &= D(DFT_n(S), DFT_n(Q)) \end{aligned}$$

Aangezien $k < n$ is het tevens duidelijk dat er geldt dat

$$\begin{aligned} D(DFT_n(S), DFT_n(Q)) &= \left(\sum_{f=0}^{n-1} |s_f - q_f|^2\right)^{\frac{1}{2}} \geq \left(\sum_{f=0}^{k-1} |s_f - q_f|^2\right)^{\frac{1}{2}} \\ &= D(DFT_k(S), DFT_k(Q)) \end{aligned}$$

Aangezien geldt dat

$$D(S,Q) = D(DFT_n(S), DFT_n(Q)) \geq D(DFT_k(S), DFT_k(Q))$$

kunnen we dus concluderen dat de discrete Fourier transformatie wel degelijk voldoet aan lemma 7.1. Omdat de Euclidische afstand bewaard blijft onder het toepassen van de DFT, kunnen we simpelweg de Euclidische afstandsmeting gebruiken voor de $D_{reductie}()$-meting bij de GEMINI-methode. M.a.w. zullen we dus voor de range query's in de R-Tree, zoals besproken in sectie 5.2.2, beroep doen op de Euclidische afstandsmeting.

8.1.3 Fast Fourier Transformatie

De berekening van de discrete Fourier transformatie aan de hand van de geziene formules zou $O(n^2)$ tijd vergen. Er bestaat echter een algoritme waarmee de DFT in $O(n \log n)$ tijd berekend kan worden. Dit algoritme noemt men de *fast Fourier transformatie (FFT)*. Van dit algoritme zijn er

[1] Om de geldigheid van deze gelijkheid te verklaren, baseren we ons op de stelling van Parseval (stelling 8.1).

vele varianten gekend. We bespreken het allereerste algoritme hiertoe, nl. het Danielson-Lanczos algoritme ([DL42]). Een nadeel van dit algoritme is dat het enkel een input van lengte n kan verwerken, waarbij de waarde n een macht van twee is.

Om tot een efficiënter algoritme te komen, worden de regelmatigheden in de $exp\left(\frac{-j2\pi if}{n}\right)$-functie uitgebuit. We behandelen nu het idee achter dit algoritme. We vertrekken hierbij van de originele formule van de DFT, waarbij we de constante factor $\frac{1}{\sqrt{n}}$ voor de eenvoud buiten beschouwing laten. Dit komt dus met andere woorden neer op

$$s_f = \sum_{i=0}^{n-1} S_i exp\left(\frac{-j2\pi if}{n}\right) \tag{8.1}$$

In deze formule maken we als volgt een opsplitsing tussen de even en oneven geïndexeerde elementen in de tijdreeks S ($|S|$=n)

$$s_f = \sum_{k=0}^{n/2-1} S_{2k} exp\left(\frac{-j2\pi(2k)f}{n}\right) + \sum_{l=0}^{n/2-1} S_{2l+1} exp\left(\frac{-j2\pi(2l+1)f}{n}\right)$$

We werken dit verder uit, waarbij we $n/2$ vervangen door N

$$s_f = \sum_{k=0}^{N-1} S_{2k} exp\left(\frac{-j2\pi kf}{N}\right) + exp\left(\frac{-j2\pi f}{N}\right) \sum_{l=0}^{N-1} S_{2l+1} exp\left(\frac{-j2\pi lf}{N}\right)$$

Wanneer we S_{2k} vervangen door S'_k en S_{2l+1} door S''_k, waarbij $S_{2k} = S'_k$ en $S_{2l} = S''_l$, bekomen we

$$s_f = \sum_{k=0}^{N-1} S'_k exp\left(\frac{-j2\pi kf}{N}\right) + exp\left(\frac{-j2\pi f}{N}\right) \sum_{l=0}^{N-1} S''_l exp\left(\frac{-j2\pi lf}{N}\right)$$

Hierop passen we nu de originele formule (8.1) toe

$$s_f = s'_f + exp\left(\frac{-j2\pi f}{N}\right) s''_f$$

Voor het berekenen van s_f wordt er nu dus een opdeling gemaakt in s'_f en s''_f. Hierbij reikt s'_f over al de even geïndexeerde elementen van de tijdreeks S ($|S| = n$). Bij s''_f daarentegen worden de oneven geïndexeerde elementen in rekening gebracht. Zowel s'_f als s''_f worden dus toegepast over een bereik van $n/2$. Maar het bereik van f reikt nog steeds over n. We gaan dit laatste bereik echter proberen te halveren door gebruik te maken van de eerder vermelde regelmatigheden van de $exp\left(\frac{-j2\pi if}{n}\right)$-functie. Voor deze functie geldt er namelijk dat

$$exp\left(\frac{-j2\pi(f+N)k}{N}\right) = -exp\left(\frac{-j2\pi(f)k}{N}\right)$$

Door van deze gelijkheid gebruik te maken kunnen we s_{f+N} ($=s_{f+n/2}$) uitdrukken als

$$s_{f+N} = s'_f \text{ - } exp\left(\frac{-j2\pi \mathbf{f}}{N}\right) s''_f$$

Voor s_f en s_{f+N} bekomen we dus de volgende vergelijkingen

$$s_f = s'_f + exp\left(\frac{-j2\pi f}{N}\right) s''_f$$
$$s_{f+N} = s'_f - exp\left(\frac{-j2\pi f}{N}\right) s''_f$$

Uit deze vergelijkingen blijkt duidelijk dat we bij de berekening van s_f er gratis de berekening van s_{f+N} bij krijgen. Hierdoor kan dus, zoals gewenst, het bereik over f gehalveerd worden. Door deze methode ook recursief toe te passen op s'_f en s''_f bekomen we een algoritme dat voor het berekenen van de DFT van een tijdreeks S ($|S| = n$) slechts $O(n \log n)$ tijd vergt.

Voor de implementatie van de FFT hebben we ons gedeeltelijk gebaseerd op de code uit [PBFV88].

8.2 Discrete Wavelet Transformatie

Een wavelet transformatie wordt gebruikt om functies in de tijd op efficiënte wijze voor te stellen. Vermits een tijdreeks kan gezien worden als een functie in de tijd, is het dus ook mogelijk om hiermee tijdreeksen te representeren. We behandelen hier de familie van discrete wavelet transformaties (DWT). Deze familie is een deelverzameling van de grote familie van wavelet transformaties. Voor de behandeling van deze sectie baseren we ons op [Chu92]. Alvorens hier verder op in te gaan, zullen we eerst de nodige basiskennis behandelen.

8.2.1 Basisbegrippen

We noemen $L^2(\mathbb{R})$ de vectorruimte van de kwadratisch integreerbare functies in $\mathbb{R}$. We zeggen dat een functie $f(x)$ *kwadratisch integreerbaar* is als $\int_{-\infty}^{+\infty} |f(x)|^2 dx < \infty$. Dit betekent dat iedere functie in $L^2(\mathbb{R})$ naar de waarde nul streeft bij $\pm\infty$. Het is in de $L^2(\mathbb{R})$-ruimte tevens mogelijk om het *inproduct* tussen twee functies $f(x)$ en $g(x)$ te definiëren, namelijk $\langle f(x), g(x)\rangle = \int f(x)\, g(x)\, dx$. Vermits we hier te maken hebben met kwadratisch integreerbare functies zal het inproduct nooit in ∞ resulteren en dus steeds een eindig resultaat geven. De *norm* of *energie* van een functie $f(x)$ wordt gegegeven door $\|f(x)\| = \sqrt{\langle f(x), f(x)\rangle}$.

We noemen twee functies $f(x)$ en $g(x)$ *orthogonaal* als $\langle f(x), g(x)\rangle = 0$. Beide functies zijn dan onderling onafhankelijk van elkaar. Indien twee functies $f(x)$ en $g(x)$ orthogonaal zijn en er bijkomend geldt dat $\|f(x)\| = 0$ en $\|g(x)\| = 0$, dan spreken we van *orthonormale* functies.

Een verzameling van functies in de $L^2(\mathbb{R})$-ruimte wordt een *basis* genoemd als eender welke functie in deze ruimte kan beschreven worden als een linaire

combinatie van deze basisfuncties. In het geval we te maken hebben met een verzameling van orthogonale functies spreken we van een *orthogonale basis*. Bij een basis bestaande uit orthonormale functies wordt er dan uiteraard gesproken van een *orthonormale basis*.

8.2.2 Theorie

Bij de behandeling van DFT hebben we gemerkt dat er gebruik wordt gemaakt van een sinusoïdale functie als basis om de tijdreeks te beschrijven. Bij de DWT daarentegen wordt er beroep gedaan op een recursieve functie ψ uit de $L^2(\mathbb{R})$-ruimte. Zowel de cosinus- als de sinusfunctie behoren niet tot deze ruimte. De functie ψ, ook wel de *moederwavelet* genoemd, wordt beschreven door

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k)$$

waarbij 2^j de *scaling* van x en $2^{-j}k$ de *translatie* van x vormen. De factor $2^{j/2}$ zorgt ervoor dat de norm van de moederwavelet bewaard blijft onder de verschillende scalings. Hoe we tot voorgaande gelijkheid komen, wordt besproken aan de hand van de volgende stappen:

- We hebben reeds gezien dat functies in de $L^2(\mathbb{R})$-ruimte streven naar de waarde nul bij $\pm\infty$. De vraag is nu hoe we het volledige bereik van een tijdreeks in $\mathbb{R}$ kunnen bedekken in het geval dat de moederwavelet vrij snel streeft naar de waarde nul. Om dit probleem op te lossen wordt er beroep gedaan op translaties of verschuivingen. Dergelijke translatie van de moederwavelet wordt beschreven door $\psi(x - \mathbf{k})$, waarbij $k \in \mathbb{Z}$. Deel (a) van figuur 8.4 verduidelijkt het begrip translatie.

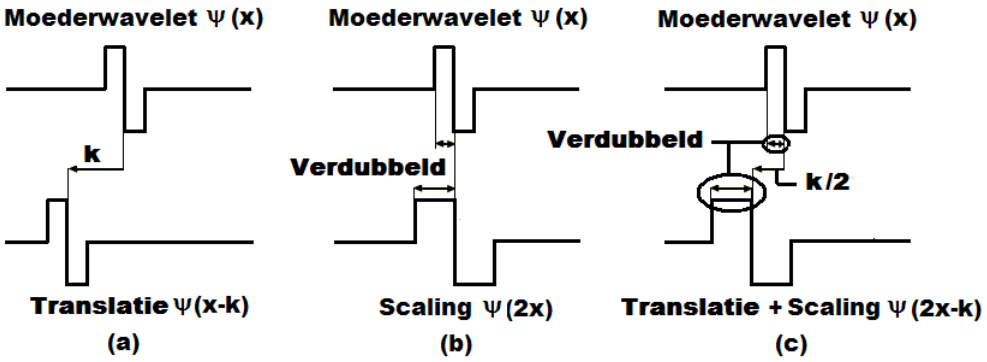
- Bij DFT wordt een tijdreeks voorgesteld door een samenstelling van sinusoïdale functies van verschillende frequenties. Bij DWT wordt dit verwezenlijkt m.b.v. scaling. Deze scaling wordt beschreven door $\psi\left(\mathbf{2^j}x\right)$, waarbij $j \in \mathbb{Z}$. Om een efficiënte berekenbaarheid te bekomen, gebeurt deze scaling steeds in machten van twee. Het begrip scaling wordt geïllustreerd in deel (b) van figuur 8.4.

- De factor $2^{j/2}$ bekomen we door de norm van $\psi(2^j x - k)$ in de $L^2(\mathbb{R})$-ruimte te berekenen, namelijk

  $$\|\psi(2^j x - k)\| = \sqrt{\int_{-\infty}^{+\infty} |f(2^j x - k)|^2 \, dx} = \mathbf{2^{-j/2}} \|\psi(x)\|$$

  waarbij $j, k \in \mathbb{Z}$.

Deel (c) van figuur 8.4 schetst de volledige situatie in het specifieke geval dat $j = 1$.

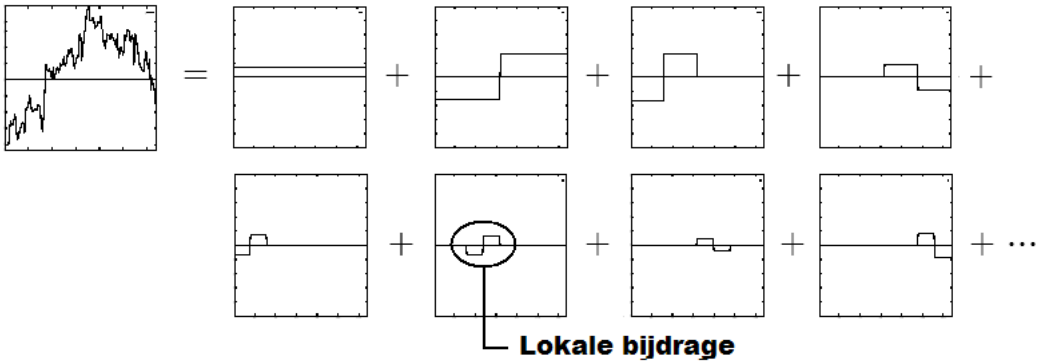


Figuur 8.4: De moederwavelet.

Elke functie $f(x)$, en dus ook elke tijdreeks, in de $L^2(\mathbb{R})$-ruimte kan nu als volgt beschreven worden

$$f(x) = \sum_{j,k} a_{j,k}\psi_{j,k}(x) = \sum_{j,k} a_{j,k} 2^{j/2}\psi(2^j x - k)$$

waarbij $a_{i,k}$ de *wavelet coëfficiënten* voorstellen. In tegenstelling tot de Fourier coëfficiënten leveren de wavelet coëfficiënten echter geen bijdrage tot de representatie van een functie over de gehele lengte van deze representatie. Dergelijke coëfficiënten leveren slechts enkel een lokale bijdrage. Figuur 8.5 verduidelijkt dit verschil tussen DWT en DFT.



Figuur 8.5: Benadering van de originele sequentie m.b.v. een DWT. [WAA00]

In de familie van de discrete wavelet transformaties kunnen we vele soorten transformaties onderscheiden. Het verschil tussen deze transformaties situeert zich in hun moederwavelets. Elke soort DWT maakt immers gebruik van een ander moederwavelet. In de volgende sectie behandelen we een bepaalde DWT, die reeds veel gebruikt werd met betrekking tot similarity search, namelijk de Haar wavelet transformatie.

8.2.3 Haar wavelet transformatie

Bij de *Haar wavelet transformatie* wordt er beroep gedaan op een vrij eenvoudige moederwavelet. Deze moederwavelet wordt ook wel eens de *Haar wavelet* genoemd en wordt beschreven door

$$\psi(x)= \phi(2x)\text{-}\phi(2x-1)$$

waarbij

$$\phi(x) = \begin{cases} 1 & 0 < x < 1 \\ 0 & \text{Voor alle andere } x\text{-waarden} \end{cases}$$

De Haar wavelet wordt m.a.w. expliciet weergegeven door

$$\psi(x) = \begin{cases} 1 & 0 < x < 0.5 \\ -1 & 0.5 < x < 1 \\ 0 & \text{Voor alle andere } x\text{-waarden} \end{cases}$$

De *(Haar) wavelet coëfficiënten* $a_{j,k}$ kunnen vervolgens berekend worden uit het inproduct $\langle \psi_{j,k}(x), f(x) \rangle$, wegens (zoals eerder gezien)

$$f(x)=\sum_{j,k} a_{j,k}\psi_{j,k}(x)= \sum_{j,k} a_{j,k} 2^{j/2}\psi(2^j x - k)$$

Deze coëfficiënten kunnen tevens gevonden worden d.m.v. een opeenvolging van matrixvermenigvuldigingen van de vorm

$$\begin{bmatrix} x'_0 \\ c'_0 \\ x'_1 \\ c'_1 \\ \vdots \\ x'_{n/2-1} \\ c'_{n/2-1} \\ x'_{n/2} \\ c'_{n/2} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & & & & \\ 1 & -1 & 0 & 0 & & & & & \\ 0 & 0 & 1 & 1 & & & & & \\ 0 & 0 & 1 & -1 & & & & & \\ \vdots & & & & \ddots & & & & \vdots \\ & & & & & 1 & 1 & 0 & 0 \\ & & & & & 1 & -1 & 0 & 0 \\ & & & & & 0 & 0 & 1 & 1 \\ & & & & \cdots & 0 & 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-3} \\ x_{n-2} \\ x_{n-1} \\ x_n \end{bmatrix} \tag{8.2}$$

De werkwijze van deze methode van achtereenvolgende matrixvermenigvuldigingen wordt beschreven aan de hand van de volgende stappen:

- De tijdfunctie of tijdreeks (van lengte $n+1$) wordt omgezet naar een kolommatrix $[x_0 \cdots x_n]^T$. Deze kolommatrix wordt dan gebruikt als input voor de matrixvermenigvuldiging uit (8.2). Dit geeft de kolommatrix $\left[x'_0\ c'_0\ \cdots x'_{n/2}\ c'_{n/2}\right]^T$ als resultaat .

- De resulterende kolommatrix $\left[x'_0\ c'_0\ \cdots x'_{n/2}\ c'_{n/2}\right]^T$ uit de vorige stap wordt gebruikt om een nieuwe kolommatrix mee op te stellen. De nieuwe matrix is van de vorm $\left[x'_0\ \cdots\ x'_{n/2}\ 0\ \cdots\ 0\right]^T$. Hierbij zijn o.a. alle c'_i-waarde vervangen door nul-waarden. We zouden dit kunnen opvatten alsof de kolommatrix uit vorige stap "gehalveerd" werd. Deze bekomen kolommatrix wordt dan op haar beurt opnieuw onderworpen aan de matrixvermenigvuldiging uit (8.2). De c'_i-waarden worden echter wel bijgehouden omdat deze, de Haar wavelet coëfficiënten voorstellen. Deze stap wordt recursief herhaald totdat we de matrixvermenigvuldiging uit (8.2) in het totaal $log_2 n$ keer uitgevoerd hebben. Dit resulteert in een kolommatrix van de vorm $\left[x_0'^{\cdots\prime}\ c_0'^{\cdots\prime}\ 0\ \cdots\ 0\right]^T$.

Zoals in de voorgaande stappen vermeld wordt, wordt de originele kolommatrix in elke stap steeds gehalveerd. Het is dan ook niet verwonderlijk dat Haar wavelet transformaties enkel werken op inputs van lengte l, waarbij de waarde van l een macht van twee voorstelt. We verduidelijken de voorgaande stappen nu aan de hand van een voorbeeld.

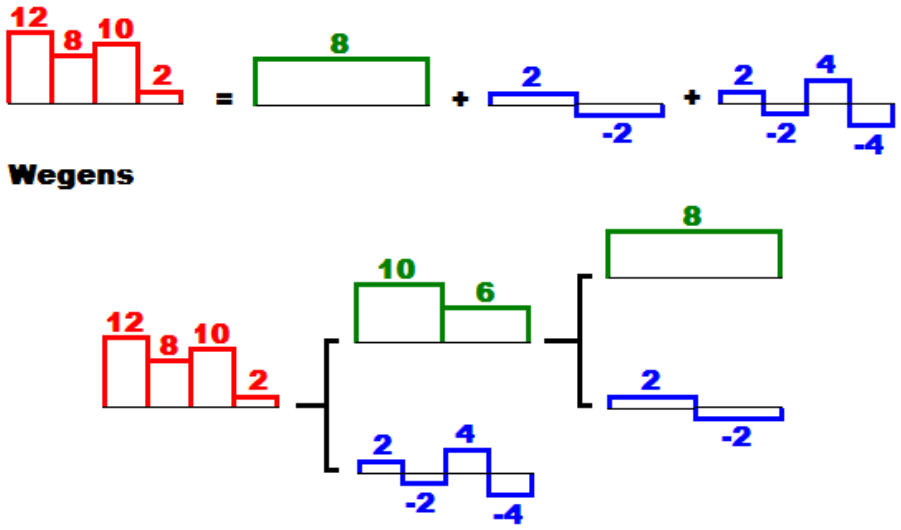
Voorbeeld 8.1 De Haar wavelet coëfficiënten $\{w_0, \ldots, w_{n-1}\}$ van tijdreeks $\{12, 8, 10, 2\}$ worden gegeven door $\{8, 2, 2, 4\}$. De tijdreeks is van lengte vier, bijgevolg zijn dus twee ($= log_2\ 4$) stappen nodig om deze coëfficiënten te berekenen. Bij de eerste stap passen we de matrixvermenigvuldiging uit (8.2) toe op de tijdreeks. Deze stap wordt gegeven door (coëfficiënten zijn onderlijnd)

$$\frac{1}{2}\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & -1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 1 & -1\end{bmatrix}\begin{bmatrix}12\\ 8\\ 10\\ 2\end{bmatrix} = \begin{bmatrix}10\\ \underline{2}\,(= w_2)\\ 6\\ \underline{4}\,(= w_3)\end{bmatrix}$$

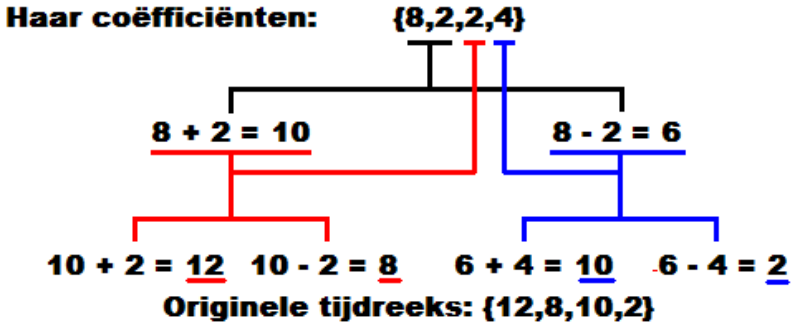
Voor de tweede stap zetten we de bekomen kolommatrix $[10\ 2\ 6\ 4]^T$ om naar de nieuwe kolommatrix $[10\ 6\ 0\ 0]^T$ en passen vervolgens opnieuw de matrixvermenigvuldiging uit (8.2) toe

$$\frac{1}{2}\begin{bmatrix}1 & 1 & 0 & 0\\ 1 & -1 & 0 & 0\\ 0 & 0 & 1 & 1\\ 0 & 0 & 1 & -1\end{bmatrix}\begin{bmatrix}10\\ 6\\ 0\\ 0\end{bmatrix} = \begin{bmatrix}\underline{8}\,(= w_0)\\ \underline{2}\,(= w_1)\\ 0\\ 0\end{bmatrix}$$

Figuur 8.6 vormt de grafische weergave van dit voorbeeld. Het is tevens mogelijk om de originele tijdreeksen af te leiden uit de gevonden coëfficiënten. Dit wordt geïllustreerd aan de hand van figuur 8.7.



Figuur 8.6: Voorbeeld van de Haar wavelet transformatie



Figuur 8.7: Omzetting van Haar coëfficiënten naar originele tijdreeks

De complexiteit van de Haar wavelet transformatie wordt gegeven door lemma 8.1. Het bewijs van dit lemma vinden we terug in [CF99].

Lemma 8.1 ([CF99]) *Gegeven een sequentie van lengte n, waarbij de waarde van n een macht van twee is, dan wordt de* complexiteit *van de Haar wavelet transformatie gegeven door O(n).*

De Euclidische afstand tussen twee tijdreeksen kan berekend worden aan de hand van hun Haar wavelet coëfficiënten volgens de methode gegeven in lemma 8.2. Het bewijs van de correctheid van deze methode wordt gegeven in [CFY03].

Lemma 8.2 ([CFY03]) *Gegeven twee sequenties X en Y (allebei van lengte n) en hun overeenkomstige Haar wavelet coëfficiënten $\left\{w_0^X, \ldots, w_{n-1}^X\right\}$ en $\left\{w_0^Y, \ldots, w_{n-1}^Y\right\}$, dan kan de Euclidische afstand tussen X en Y als volgt in termen van $\left\{w_0 \left(= w_0^X - w_0^Y\right), \ldots, w_{n-1} \left(= w_{n-1}^X - w_{n-1}^Y\right)\right\}$ uitgedrukt worden*

$$D(X,Y) = S_{log_2\, n}$$

waarbij voor $0 \leq i \leq log_2 n - 1$

$$S_{i+1} = \sqrt{2}\left(S_i^2 + w_{2^i}^2 + w_{2^i+1}^2 + \ldots + w_{2^{i+1}-1}^2\right)$$
$$S_0 = w_0$$

Voor het reduceren van de dimensionaliteit van een tijdreeks van lengte n maken we, net als bij DFT, enkel gebruik van de eerste k wavelet coëfficiënten. In [CFY03] wordt voorgesteld om de normalisatiefactor $\frac{1}{2}$ in de matrixvermenigvuldiging uit (8.2) te vervangen door $\frac{1}{\sqrt{2}}$. Op deze wijze kan de Euclidische afstand tussen de sequentie X en Y uit lemma 8.2 als volgt berekend worden

$$D(X,Y) = \left(\left(w_0^X - w_0^Y\right)^2 + \ldots + \left(w_{n-1}^X - w_{n-1}^Y\right)^2\right)^{1/2}$$

Hierin herkennen we duidelijk de formule van de Euclidische afstandsmeting (zie definitie 6.1). Door m.a.w. te werken met normalisatiefactor $\frac{1}{\sqrt{2}}$ blijft de Euclidische afstand bewaard onder de Haar wavelet transformatie. Met betrekking tot de range query's bij een SAM (en in het bijzonder de R-Tree) zullen we dan ook eenvoudigweg beroep doen op de Euclidische afstandsmeting. De garantie dat er zich bij de GEMINI-methode geen false dismissals voordoen door enkel de k eerste coëfficiënten in rekening te brengen bij de juist besproken afstandsmeting wordt gegeven door lemma 8.3. Het bewijs van dit lemma vinden we terug in [CFY03].

Lemma 8.3 ([CFY03]) *Door bij de Haar wavelet transformatie met voorgaande normalisatiefactor ($\frac{1}{\sqrt{2}}$) enkel gebruik te maken van de k ($1 \leq k \leq n$) eerste coëfficiënten, zullen er zich geen false dismissals voordoen bij een range query.*

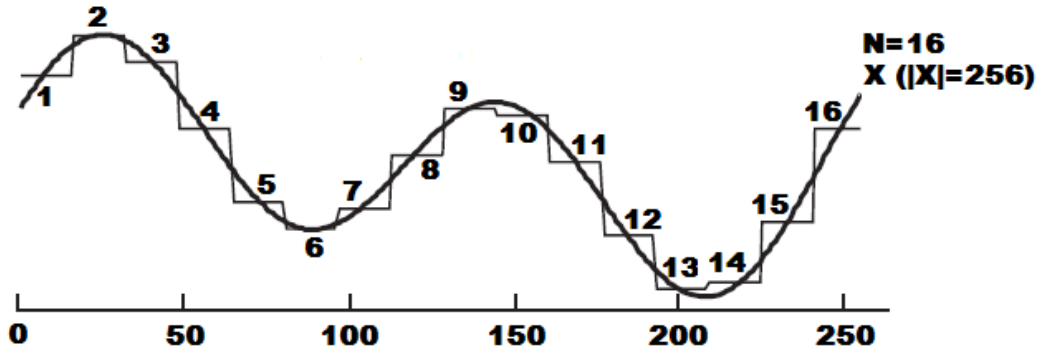
8.3 Piecewise Aggregate Approximation

Een laatste reductietechniek die we in beschouwing nemen is bekend onder de naam *Piecewise Aggregate Approximation*(*PAA*). Deze techniek werd voorgesteld in [KCPM01].

Om een sequentie $S(=s_1, \ldots, s_n)$ van lengte n om te zetten naar een punt $S'(= s'_1, \ldots, s'_N)$ in de N-dimensionale ruimte ($N \leq n$) kunnen we beroep doen op de volgende formule

$$s'_i = \frac{N}{n} \sum_{j=\frac{n}{N}(i-1)+1}^{\frac{n}{N}i} s_j$$

In het geval dat $N = 1$ wordt er met behulp van deze formule gewoon het gemiddelde berekend van de waarden van de sequentie S. Bij $N = n$ behouden we gewoon de waarden van de sequentie S, m.a.w. $(s'_1 (= s_1), \ldots, s'_N (= s_n))$. Voor de overige N-waarden ($1 < N < n$) wordt de sequentie S gelijkmatig opgesplitst in N frames van dezelfde lengte. Vervolgens wordt dan simpelweg het gemiddelde berekend van elk frame. Figuur 8.8 illustreert hoe een sequentie X, van lengte 256, met de PAA-techniek benaderd ($N = 16$) wordt. We verduidelijken deze techniek tevens a.d.v. voorbeeld 8.2.



Figuur 8.8: Voorbeeld van de PAA-techniek.

Voorbeeld 8.2 Voor het reduceren van sequentie $S(= 8, -2, 4, 0, 4, 4, -2, -4)$ (van lengte 8) naar een 4-dimensionaal punt $S'(= s'_1, \ldots, s'_4)$ gaan we als volgt te werk

$$\frac{N}{n} = \frac{4}{8} = \frac{1}{2} \Rightarrow \text{frames van lengte twee.}$$

$$s'_1 = \tfrac{1}{2}(8-2) = 3,\ s'_2 = \tfrac{1}{2}(4+0) = 2,\ s'_3 = \tfrac{1}{2}(4+4) = 4,\ s'_4 = \tfrac{1}{2}(-2-4) = -3$$

Dit resulteert dus in $S'(s'_1 = 3, s'_2 = 2, s'_3 = 4, s'_4 = -3)$.

De tijdscomplexiteit van deze methode is $O(n)$, waarbij n de lengte is van de sequentie. Met het oog op subsequence matching kan het zijn dat een sequentie opgesplitst wordt in w aantal sliding windows van lengte m. Elke window wordt dan onderworpen aan de PAA-methode om de dimensionaliteit van dat window te reduceren naar N. In dat geval vereist deze

methode $O(wm)$-tijd om dit voor alle sequenties te verwezenlijken. We hebben echter reeds gezien dat het verschil tussen twee opeenvolgende sliding windows niet zo groot is. Dit werd reeds geïllustreerd in figuur 4.5. Door op dit klein verschil in te spelen kunnen we de benodigde tijd om dit alles te berekenen gevoelig verbeteren, nl. $O(Nm)$-tijd. Voor de berekening van het N-dimensionale punt van window j wordt er beroep gedaan op het reeds berekende N-dimensionale punt van window $(j-1)$. Dit wordt verwezenlijkt m.b.v. de volgende formule

$$s'_{j,i} = s'_{j-1,i} - \frac{N}{n} s_{\frac{n}{N}(i-1)+1} + \frac{N}{n} s_{\left(\frac{n}{N}i\right)+1}$$

Zij Q en S twee sequenties van lengte n en zij Q' en S' respectievelijk hun overeenkomstige N-dimensionale punten, dan kunnen we de afstand tussen Q' en S' (volgens [KCPM01]) als volgt berekenen

$$D_{PAA}(Q',S') = \sqrt{\frac{n}{N}} \sqrt{\sum_{i=1}^{N} (q'_i - s'_i)^2}$$

In [KCPM01] wordt tevens bewezen dat er geldt dat $D_{PAA}(Q',S') \leq D(Q,S)$. Met deze eigenschap zijn we er dan ook meteen van verzekerd dat er zich bij de GEMINI-methode geen false dismissals zullen voordoen. Een laatste opmerking die we bij deze $D_{PAA}()$-afstandsfunctie maken is, dat we hier beroep doen op de $D_{reductie}()$-meting bij de GEMINI-methode.

8.4 Overzicht reductietechnieken

Tot slot van dit hoofstuk aanschouwen we nog kort tabel 8.1, waar er een overzicht gegeven wordt van de behandelde reductietechnieken. Naast de naam van de techniek wordt hierbij tevens aangegeven welke afstandsmeting er zal gebruikt worden voor de $D_{reductie}()$-meting bij de GEMINI-methode.

Tabel 8.1: Overzicht reductietechnieken

Naam	$\mathbf{D_{reductie}}()$
DFT	Euclidische afstandsmeting
DWT (Haar)	Euclidische afstandsmeting
PAA	$D_{PAA}()$

Hoofdstuk 9

Testresultaten: GEMINI

In dit hoofdstuk bespreken we de testresultaten van onze implementatie van de GEMINI-methode. Bij het testen van deze methode hebben we beroep gedaan op de drie reductietechnieken uit het vorige hoofdstuk. We zullen hiervan dan ook een korte bespreking geven. We vatten dit hoofdstuk echter aan met een methode waarbij de datasequenties onderworpen werden aan een sequentiële scan. Aan de hand van de resultaten van deze scan zal de suprematie van de GEMINI-methode aangetoond worden.

Al het testwerk werd verricht op de ECG-sequenties uit de MIT-BIH Arrhythmia Database ([Res]). We gebruiken echter niet alle beschikbare sequenties in deze database. Voor ons testwerk hebben we steeds gewerkt met een database bestaande uit 10144 datasequenties, elk van lengte 1024. Deze sequenties werden bekomen door enkele (zeer lange) sequenties uit de MIT-BIH Arrhythmia Database in stukken van lengte 1024 te knippen. Voor de querysequenties werkten we steeds met sequenties van lengte 128. We hebben hiervoor twee soorten van querysequenties gebruikt. De eerste soort zijn querysequenties die we gewoon uit één van de datasequenties geknipt hebben. De datasequentie, waaruit een dergelijke querysequentie geknipt wordt, bevat dus minstens één deelsequentie die volledig identiek is met deze querysequentie. De andere soort querysequenties zijn bekomen door ze uit één van de datasequenties te knippen. Hierna hebben we deze querysequenties echter manueel een heel klein beetje aangepast, zodat de resulterende sequentie een beetje afwijkt van het originele uitgeknipte deel.

Het testwerk werd verricht op een Intel Dual Core P4 3.20GHz processor met 1024 MB geheugen en 80 GB opslagruimte.

9.1 Sequentiële scan

Bij deze methode wordt elke mogelijke deelsequentie van iedere datasequentie in de database vergeleken met de querysequentie. De datasequenties worden m.a.w. sequentieel doorlopen. Vandaar dat we spreken van een sequentiële scan. Het vergelijken gebeurt, net als bij de GEMINI-methode, op basis van Euclidische afstandsmeting. In deze methode wordt er echter geen SAM gebruikt. Het opzet van deze methode is dan ook het nut van een SAM duidelijk te maken. Vermits elke mogelijke deelsequentie van iedere datasequentie vergeleken wordt met de querysequentie, is het uiteraard duidelijk dat er zich geen false dismissals kunnen voordoen. We hebben deze methode getest voor querysequenties van lengte 128 met een tolerantie $\epsilon = 0,5$. Voor onze database met 10144 datasequenties geeft dit een totale zoektijd van om en bij de 27 minuten. We zullen zien dat deze zoektijden m.b.v. de GEMINI-methode gevoelig verminderd kunnen worden. Voor de volgende methoden zullen we steeds werken met dezelfde tolerantie, nl. $\epsilon = 0,5$.

9.2 Sliding windows

We hebben er in dit werk voor gekozen om de General Match methode te gebruiken omdat deze de voordelen van FRM en Dual Match combineert. In ons testwerk hebben we dan ook een klein beetje geëxperimenteerd met de waarde van J en de grootte van de windows. Hoe kleiner we J nemen, hoe groter we onze maximale windowgrootte kunnen nemen. Zoals we gemerkt hebben, brengen grotere windows minder false alarms met zich mee wegens het window size effect. Maar een kleinere waarde voor J brengt ook meer windows afkomstig van de datasequenties met zich mee. We hebben dit getest voor $J = 4$, $J = 16$ en $J = 32$, waarbij we dan steeds (volgens lemma 4.9) beroep deden op de grootst mogelijke windows. Ondanks het feit dat kleinere J-waarden, wegens het windows size effect, de neiging hebben om minder false alarms te genereren, gaf de situatie voor $J = 32$ minder false alarms dan voor $J = 4$. Dit is waarschijnlijk te wijten aan het feit dat er bij $J = 32$ veel minder windows, afkomstig van de datasequenties, aanwezig waren om mee te matchen. Het verschil tussen de situaties waarbij $J = 16$ en deze waarbij $J = 32$ was echter niet zo fel merkbaar. We raden dan ook aan in deze situatie te kiezen voor $J = 16$ of $J = 32$. Door op dit vlak nog wat meer testwerk te verrichten, kan er naar een optimale waarde voor J gezocht worden.

9.3 Fast Fourier transformatie

We bespreken nu onze testresultaten van de GEMINI-methode, waarbij we de fast Fourier transformatie als reductietechniek gebruikt hebben. We hebben dit getest zowel in het geval dat de vier Fourier coëfficiënten gebruikt worden, als in het geval dat er beroep gedaan wordt op de eerste acht coëfficiënten. De situatie waarbij we een beroep deden op de eerste acht coëfficiënten gaf ons veel betere resultaten. In deze situatie bevatte de kandidatenverzameling bij onze tests tot 25% minder windows. Dit levert uiteraard een enorme tijdwinst op, vermits er veel minder false alarms uit de kandidatenverzameling moeten verwijderd worden. Hieruit blijkt dus duidelijk dat de eerste acht Fourier coëfficiënten de originele windows toch opmerkelijk beter beschrijven dan alleen de eerste vier coëfficiënten.

Waar de sequentiële scan om en bij de 27 minuten nodig had om alle matchings te vinden, vergde het bij deze manier van werken (gemiddeld gezien) ongeveer 8 minuten. Dit benadrukt de kracht van het gebruik van een SAM. Voor onze tests hebbben we een beroep gedaan op de R-Tree. In de tweede stap van de GEMINI-methode werd er tot ongeveer 80% van de windows uit de R-Tree weggesnoeid. Hoe goed een R-Tree wordt gesnoeid, hangt uiteraard samen met welke tolerantie ϵ er gebruikt wordt. Hoe kleiner ϵ, hoe meer takken er gesnoeid zullen worden in de R-Tree. Maar indien we een te kleine waarde nemen voor ϵ, zullen echter alle takken gesnoeid worden. In dat geval zullen er dus geen resultaten gevonden worden. Dit geldt tevens voor de hierop volgende technieken en we zullen dat dan ook niet meer herhalen.

9.4 Haar wavelet transformatie

Voor de GEMINI-methode, waarbij we beroep doen op de Haar wavelet transformatie als reductietechniek, hebben we ook het verschil getest tussen het gebruik van de eerste vier en de eerste acht coëfficiënten. Net als bij de fast Fourier transformatie bleken de eerste acht coëfficiënten de windows veel beter te beschrijven dan de eerste vier. Dit bleek uit het feit dat de kandidatenverzameling tot 27% minder windows bevatte in de situatie waarbij de eerste acht coëfficiënten gebruikt werden.

Deze manier van werken vergde gemiddeld genomen ongeveer 7 minuten tijd. Ook hier is dit verschil t.o.v. de sequentiële scan weer te danken aan het gebruik van de R-Tree. Bij het snoeien van de R-Tree werd er onderweg soms tot 83% van de windows in de R-Tree weggesnoeid. Op dit vlak scoort deze methode iets beter dan in het geval de fast Fourier transformatie gebruikt

werd als reductietechniek. Dit zou te wijten kunnen zijn aan het feit dat de Haar wavelet coëfficiënten de ECG-sequenties misschien net iets beter beschrijven dan de Fourier coëfficiënten.

9.5 Piecewise Aggregate Approximation

Tot slot hebben we ook de GEMINI-methode getest, waar de PAA-techniek gebruikt wordt voor het reduceren van de dimensionaliteit van de windows. Deze methode kwam als beste uit ons testwerk. Het verschil tussen het reduceren van de dimensionaliteit van de windows naar vier enerzijds of naar acht anderzijds, is hier echter wel nog opmerkelijker dan in de voorgaande testen. Bij het gebruik van een 8-dimensionale voorstelling van de windows bevatte de kandidatenverzameling soms tot bijna 50% windows minder dan wanneer een 4-dimensionale voorstelling gebruikt werd.

Na het snoeien in de tweede stap van de GEMINI-methode bleef gemiddeld nog slechts een kleine 10% van het totaal aantal windows over. Dit wijst er dus op dat de ECG-sequenties vrij goed beschreven kunnen worden m.b.v. de PAA-techniek. Over het algemeen vergde deze manier van werken slechts ongeveer 5 minuten tijd. Dit houdt toch een enorme verbetering in t.o.v. de tijd nodig bij de sequentiele scan.

Hoofdstuk 10

Minimale variantie matching

In dit hoofdstuk behandelen we een volledig nieuwe methode voor subsequence matching. Deze methode wordt de minimale variantie matching methode of kortweg de MVM-methode genoemd. Het verschil met de voorgaande methoden voor subsequence matching is dat deze methode geen beroep doet op sliding window technieken. De MVM-methode is, net als de GEMINI-methode, wel nog gebaseerd op de Euclidische afstandsmeting en wordt voorgesteld in [LMW+05a] en [LMW+05b]. Vooraleer we de MVM-methode behandelen bespreken we kort enkele basisbegrippen.

10.1 Basisbegrippen

Een *gerichte acyclisch graaf* (of *DAG*: directed acyclic graph) wordt volgens [Nat] omschreven als een gerichte graaf, waarbij er geen enkel pad kan bestaan dat begint en eindigt bij dezelfde knoop. Bij de MVM-methode wordt er gewerkt met een graaf waarbij aan alle bogen een kost toegekend is. De kost van een boog van knoop A naar knoop B kan men opvatten als de kost om zich te verplaatsen van A naar B. Voor de MVM-methode hebben we nood aan een *kortste pad algoritme* dat de kosten van alle paden vanuit een opgegeven vertrekknoop berekent. We bespreken dit algoritme aan de hand van pseudocode waarbij we twee aparte functies onderscheiden, nl. de functies *KortstePad()* en *VergelijkAfstand()*. Bij de VergelijkAfstand()-functie wordt er gebruik gemaakt van de twee array's van lengte K (= het aantal knopen in de graaf), nl. afstand[] en pad[]. Elke entry van deze twee array's komt uniek overeen met één van de knopen in de graaf. Zo worden afstand[0] en pad[0] bijvoorbeeld gekoppeld aan de vertrekknoop. De knoop overeenkomstig met de i-de entry zullen we voor de eenvoud benoemen als k_i. De i-de entry van de array afstand[], nl. afstand[i], houdt de kortste

afstand bij tussen de vertrekknoop en de knoop k_i. In pad[i] wordt de knoop bewaard die voor k_i lag op het kortste pad tussen de vertrekknoop en k_i. Alle entry's van beide array's worden geïnitialiseerd op oneindig, op uitzondering van afstand[0] en pad[0]. Deze entry's worden namelijk geïnitialiseerd op de waarde nul. Dit is ook logisch daar deze entry's betrekking hebben op de vertrekknoop. De graaf wordt voorgesteld als een geordende dubbele array, nl. $g[][]$. Met geordend bedoelen we dat we op de eerste rij van de array ($g[0][i]$) alle knopen vinden op boogafstand één van de vertrekknoop, de tweede rij ($g[1][i]$) bevat vervolgens alle knopen op boogafstand twee, enz. De inhoud van deze array $g[][]$ beschrijft de boogkosten.

Pseudocode 2 KortstePad(GRAAF[][] g)

```
/* We lopen door alle knopen beginnend bij de vertrekknoop */
for (alle knopen k in g) do
  for (alle knopen l op boogafstand 1 van knoop k) do
    /* Zoek de overeenkomstige index van knoop k in de array's afstand[]
       en pad[]. */
    INT index1 = k.geefIndex();
    /* Zoek de overeenkomstige index van knoop l in de array's afstand[]
       en pad[]. */
    INT index2 = l.geefIndex();
    /* Zoek de kost van de boog tussen knoop k en knoop l. */
    INT kost = g[k][l];
    /* Hulpfunctie: zie pseudocode hieronder. */
    VergelijkAfstand(index1,index2,kost);
  end for
end for
```

Pseudocode 3 VergelijkAfstand(INT $ind1$,INT $ind2$,DOUBLE $kost$)

```
if (afstand[ind2] > afstand[ind1] + kost) then
  /* Er is een goedkoper pad gevonden om vanuit de vertrekknoop naar
     de knoop k_ind2(= l → zie hierboven) te gaan. Het nieuwe pad loopt
     dan via de knoop k_ind1(= k → zie hierboven). */
  afstand[ind2] = afstand[ind1] + kost;
  pad[ind2] = ind1;
end if
```

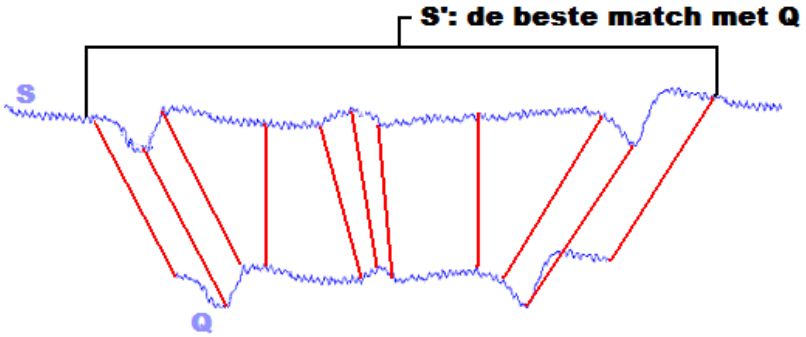
Zoals we zien wordt de tijdscomplexiteit van bovenstaand algoritme gegeven door $O(KB)$, met K het aantal knopen en B het aantal bogen.

We noemen een functie $f : \{1, \ldots, m\} \rightarrow \{1, \ldots, n\}$ een *injectie* als alle elementen van $\{1, \ldots, m\}$ een ander beeld hebben. Voor elk element y van $\{1, \ldots, n\}$ bestaat er dus hoogstens één element x van $\{1, \ldots, m\}$ zodat y het beeld is van x. We noemen een injectie een *monotoon stijgende injectie* als $f(x) < f(x+1)$.

10.2 Theorie

We beschrijven de MVM-methode aan de hand van twee sequenties $Q(=q_1, \ldots, q_m)$ en $S(=s_1, \ldots, s_n)$ van verschillende lengten, nl. $m(=|Q|)$ en $n(=|S|)$. Tevens gaan we ervan uit dat de lengte m kleiner is dan de lengte n. Bij de MVM-methode wordt naar de deelsequentie S', bevat in S, gezocht die het best matcht met de sequentie Q. Wanneer we dus een querysequentie vergelijken met k datasequenties, wordt er dus voor elk van de k datasequenties één deelsequentie teruggegeven die het best matcht met de querysequentie.

De MVM-methode kan opgevat worden als een elastische methode voor subsequence matching. Hieronder verstaan we dat we bij het zoeken van een deelsequentie van S geen rekening houden met de lengte van deze deelsequentie t.o.v. de lengte van Q. We verduidelijken dit m.b.v. figuur 10.1. Hier zien we dat sequentie Q het best matcht met een langere deelsequentie S' van S. Elk punt van Q wordt hier afgebeeld op een punt van S'. Het omgekeerde is echter niet waar.



Figuur 10.1: Voorbeeld van de MVM-methode

Het idee achter de MVM-methode is dat er naar een monotoon stijgende injectie $f : \{1, \ldots, m\} \rightarrow \{1, \ldots, n\}$ gezocht wordt, zodat $D(Q,S,f)$ een minimale waarde heeft, waarbij

$$D(Q,S,f) = \sqrt{\sum_{i=1}^{m} \left(s_{f(i)} - q_i\right)^2}$$

In principe betekent dit evenveel als het minimaliseren van de variantie $\sigma^2(Q,S,f)$ waarbij

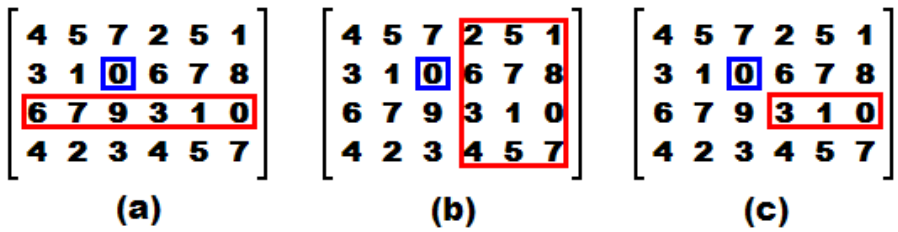
$$\sigma^2(Q,S,f)=\frac{1}{m}\sum_{i=1}^{m}\left(s_{f(i)}-q_i\right)^2$$

Dit verklaart dan ook meteen de naam van de methode, nl. minimale variantie matching.

We bespreken nu een methode voor het vinden van een injectie f om de variantie $\sigma^2(Q,S,f)$ te minimaliseren. Hiervoor bouwen we een matrix M op met n kolommen en m rijen, waarbij $M[i][j] = (s_i - q_j)$ met $(0 \leq i \leq m-1)$ en $(0 \leq j \leq n-1)$. De bekomen matrix M wordt vervolgens omgevormd tot een gerichte acyclische graaf. Elke $M[i][j]$ (met $0 \leq i \leq m-1$ en $0 \leq j \leq n-1$) wordt hierbij beschouwd als een knoop van de graaf. Er bevindt zich een boog tussen de twee knopen $M[i][j]$ en $M[k][l]$ als en slechts als deze knopen aan de volgende twee voorwaarden voldoen

$$\begin{array}{l}(1)\ k-i=1\\(2)\ j<l\end{array}$$

De eerste voorwaarde houdt in dat er zich in de matrix (of graaf) alleen rechtstreekse bogen bevinden tussen knopen afkomstig van twee opeenvolgende rijen in de matrix. Dit wordt verduidelijkt in deel (a) van figuur 10.2. Al de knopen in deze figuur die vanuit de blauwe knoop bereikbaar zijn volgens de eerste voorwaarde zijn aangeduid in het rood. D.m.v. de tweede voorwaarde wordt aangegeven dat er vanuit een knoop in de j-de kolom enkel uitgaande bogen bestaan naar knopen gelegen in een kolom rechts van deze j-de kolom. Het is dus m.a.w. onmogelijk om naar een knoop te gaan in een kolom links gelegen van de j-de kolom of naar een knoop gelegen in de j-de kolom zelf. Deel (b) van figuur 10.2 verduidelijkt de tweede voorwaarde. Ook hier zijn alle knopen die vanuit de blauwe knoop bereikbaar zijn volgens de tweede voorwaarde aangeduid in het rood. In deel (c) van figuur 10.2 wordt de combinatie van beide voorwaarden verduidelijkt. De kost van een boog tussen de knopen $M\,[i]\,[j]$ en $M\,[k]\,[l]$ wordt gegeven door $(M\,[k]\,[l])^2$.



Figuur 10.2: Geldige bogen van de graaf

Na het opstellen van de graaf, zoeken we naar het kortste (goedkoopste) pad in de graaf. Dit pad moet echter aan enkele voorwaarden voldoen. Ten eerste moet het pad beginnen in een knoop $M[0][j]$ waarbij $0 \leq j \leq n-m$. Het pad moet ten slotte eindigen in een knoop $M[m-1][j]$ waarbij $n-m \leq j \leq n-1$. Om dit pad te vinden passen wij het eerder besproken kortste pad algoritme toe op de graaf. Hiertoe voegen we bij de implementatie van deze methode echter nog twee extra knopen toe aan de graaf. Vanuit de eerste nieuwe knoop K_1 trekken we bogen naar de juist besproken beginknopen $M[0][j]$ $(0 \leq j \leq n-m)$. Aan de nieuwe bogen kennen we een kost van waarde nul toe zodat deze bogen geen invloed hebben op het zoeken van het kortste pad. De nieuwe knoop bevat dus zeker geen ingaande bogen. Op die manier creëren we ook geen nieuwe cyclus in de graaf. In de tweede nieuwe knoop K_2 laten we enkel ingaande bogen aankomen. Zo ontstaat er ook hier weer geen nieuwe cyclus. De inkomende bogen zijn afkomstig van de eindknopen $M[m-1][j]$ $(n-m \leq j \leq n-1)$ en krijgen ook nu weer een kost van waarde nul toegekend. Vervolgens passen we het kortste pad algoritme toe met de knoop K_1 als vertrekknoop. Het resulterende kortste pad kunnen we vervolgens gemakkelijk terugvinden door het pad tussen K_1 en K_2 te aanschouwen. Van dit kortste pad knippen we tot slot de knopen K_1 en K_2 af, zodat het invoegen van deze knopen teniet gedaan wordt. Het kolomnummer in de matrix van iedere knoop van het gevonden kortste pad beschrijft nu de injectie f nodig om de variantie $\sigma^2(Q,S,f)$ te minimaliseren. We verduidelijken deze werking van de MVM-methode aan de hand van het volgende voorbeeld.

Voorbeeld 10.1 ([LMW$^+$05a]) Zij Q=$\{1,2,8,6,8\}$ en S=$\{1,2,9,3,3,5,9\}$ twee sequenties van respectievelijke lengte 5 en 7. De injectie waarbij variantie $\sigma^2(Q,S,f)$ minimaal is, kan als volgt gevonden worden

$$M = \begin{bmatrix} \underline{0} & 1 & 8 & 2 & 2 & 4 & 8 \\ -1 & \underline{0} & 7 & 1 & 1 & 3 & 7 \\ -7 & -6 & \underline{1} & -5 & -5 & -3 & 1 \\ -5 & -4 & 3 & -3 & -3 & \underline{-1} & 3 \\ -7 & -6 & 1 & -5 & -5 & -3 & \underline{1} \end{bmatrix}$$

Het kortste pad werd in de matrix zelf aangeduid (onderlijnd). De injectie wordt gegeven door

$$f(0)= 0, f(1)= 1, f(2)= 2, f(3)= 5, f(4)= 6$$

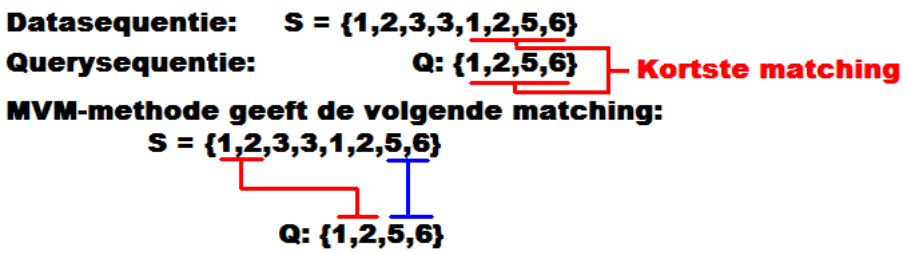
Dit leidt tot de afstand

$$D(Q,S,f)= \sqrt{(1-1)^2 + (2-2)^2 + (9-8)^2 + (5-6)^2 + (9-8)^2} = \sqrt{3}$$

Hoofdstuk 11

Testresultaten: Minimale variantie matching

We bespreken nu de testresultaten van de MVM-methode uit het vorige hoofdstuk. De tolerantie ϵ heeft totaal geen invloed op deze methode. De MVM-methode zoekt immers voor elke datasequentie S naar de deelsequentie S' (bevat in S), die het best matcht met de querysequentie Q. Wanneer we toch willen vergelijken met $\epsilon = 0,5$ (zoals bij de GEMINI-methode), zien we dat deze methode veel meer deelsequenties vindt, die minder dan een tolerantie $\epsilon = 0,5$ van de querysequentie verschillen, dan dat de GEMINI-methode zou vinden. Dit ligt, gezien de opbouw van de MVM-methode geheel binnen de verwachtingen. Met deze methode kan een querysequentie uitgerokken worden over de lengte van de datasequenties. Een probleem dat zich hierdoor met deze methode kan voordoen, is dat er een kortere matching in een datasequentie over het hoofd gezien kan worden ten koste van een langere matching in dezelfde datasequentie. Voor bepaalde domeinen zou dit misschien een voordeel kunnen zijn, maar met betrekking tot ons domein ervaren we dit toch als een nadeel. We verduidelijken dit m.b.v. het figuur 11.1.



Figuur 11.1: Nadeel MVM-methode.

Het grootste nadeel van deze methode moeten we echter nog bespreken. De tijd nodig voor het uitvoeren van deze methode is enorm in vergelijking met de GEMINI-methode. Voor onze database met 10144 datasequenties van elk van lengte 1024 en een querysequentie van lengte 128 vergt deze methode om en bij de drie en een half uur. Dit is een opmerkelijk groot tijdsverschil in vergelijking met de tijden van de GEMINI-methode. Dit komt o.a. omdat er bij deze methode geen SAM gebruikt wordt. De querysequentie wordt dus, net als bij de sequentiële scan, vergeleken met elke mogelijke deelsequeuntie van al de datasequenties. Het voordeel van het vinden van betere matchings weegt hier echter toch niet op tegen dit nadeel van lange zoektijden.

Deel II

Time Warping

Hoofdstuk 12

Dynamic Timewarping afstand

In dit tweede deel behandelen we het geval van similarity search gebaseerd op de dynamic timewarping (DTW) afstand. We vatten dit laatste deel aan m.b.v. dit hoofdstuk over de DTW-afstand. We baseren ons hierbij op [KP99] en [Keo02].

12.1 Afstandsfunctie

Een *warping-matrix* M is een $n \times m$-matrix opgesteld aan de hand van de twee sequenties $S(= s_1, \ldots, s_n)$ en $Q(= q_1, \ldots, q_m)$. We stellen het (i,j)-de element van deze matrix voor de eenvoud voor als $M[i][j]$, waarbij $1 \leq i \leq n$ en $1 \leq j \leq m$. Dergelijk element $M[i][j]$ staat voor de afstand tussen s_i en q_j. Hiervoor kunnen meerdere soorten van afstandsfuncties gebruikt worden. In dit werk zullen we voor deze afstandsmeting echter beroep doen op de Euclidische afstandsfunctie, nl. $D(s_i,q_j)=\sqrt{(s_i - q_j)^2}$. Het *warping-pad* vormt een pad in de matrix M van de vorm $W=w_1, \ldots, w_k, \ldots, w_K$, waarbij $w_k=(i, j)$ en $\max(n, m) \leq K \leq n + m - 1$. Dit pad moet tevens aan de volgende eigenschappen voldoen:

- Het warping-pad begint in $M[1][1]$. M.a.w. geldt dat w_1=(1,1).
- Het warping-pad eindigt in $M[n][m]$. M.a.w. geldt dat w_K=(n,m).
- Zij w_k =(x,y), dan is w_k =(x′,y′), waarbij $0 \leq x - x' \leq 1$ en $0 \leq y - y' \leq 1$. Dit betekent dat het pad zowel continue en als monotoon stijgend is.

Elke stap $w_k=(i, j)$ van het pad W wordt gekenmerkt door het matrixelement $M[i][j]$. We kunnen dit opvatten alsof er met elke stap $w_k=(i, j)$ een kost,

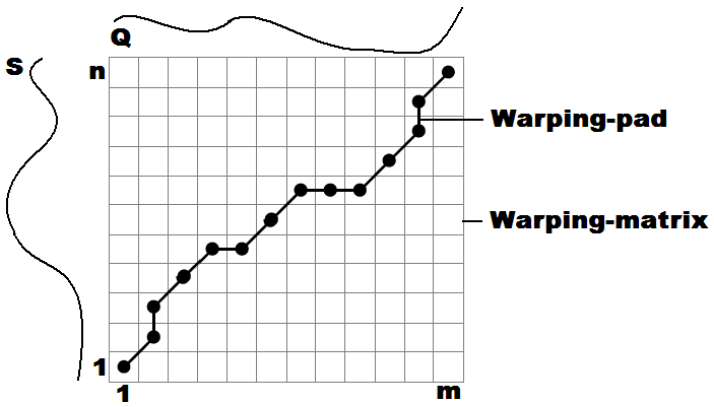
gelijk aan de inhoud van $M[i][j]$, overeenkomt. Om nu de DTW-afstand tussen de twee sequenties S en Q te berekenen zoeken we naar het pad gekenmerkt door de kleinst mogelijke totale kost, m.a.w.

$$DTW(S,Q) = min(\sqrt{\sum_{k=1}^{K} w_k})$$

In de onderstaande pseudocode wordt een algoritme ([RJ93]), gebaseerd op dynamic programming, gegeven om deze afstand te berekenen. De begrippen warping-matrix en warping-pad worden verduidelijkt m.b.v. figuur 12.1.

Pseudocode 4 BerekenDTWAfstand(SEQUENTIE S,SEQUENTIE Q)

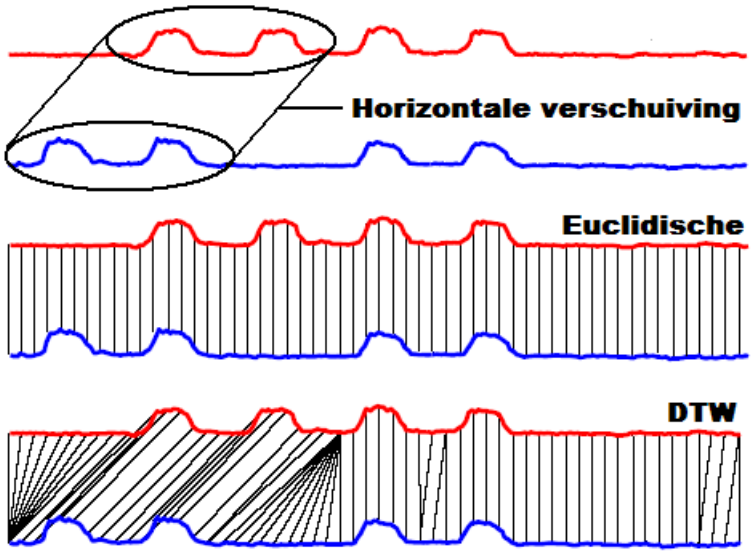
```
MATRIX kosten[][] = new MATRIX[S.lengte()][Q.lengte()];
kosten[1][1] = 0;
for (1 ≤ i ≤ S.lengte()) do
  kosten[i][1]=∞;
end for
for (1 ≤ j ≤ Q.lengte()) do
  kosten[1][j]=∞;
end for
for (1 ≤ i ≤ S.lengte()) do
  for (1 ≤ j ≤ Q.lengte()) do
    minimum = min (kosten [i] [j] , kosten [i − 1] [j] , kosten [i] [j − 1])
    kosten[i][j]= D (s_i, q_j) + minimum;
  end for
end for
return kosten[S.lengte()][S.lengte()];
```



Figuur 12.1: De begrippen warping-matrix en warping-pad. ([KP99])

Op vlak van complexiteit vergt voorgaand algoritme zowel $O(nm)$-tijd als -ruimte, waarbij $n = |S|$ en $m = |Q|$. Indien we dus werken met twee sequenties van gelijke lengte n komt dit neer op $O(n^2)$-tijd en -ruimte. Op dit vlak komt de DTW-afstandsmeting er dan ook een slechter uit dan de Euclidische afstandsfunctie ($O(n)$). Een tweede nadeel t.o.v. de Euclidische afstandsfunctie is dat DTW-afstandsmeting niet voldoet aan de driehoeksongelijkheid. Hierdoor is het dus niet mogelijk om deze afstandsfunctie te gebruiken in combinatie met bepaalde SAM's.

De DTW-afstandsmeting heeft echter niet alleen maar nadelen. In sectie 6.2 hebben we reeds een oplossing beschouwd om af te handelen met verticale verschuivingen. De Euclidische afstandsfunctie biedt echter geen oplossingen voor verschuivingen in de richting van de X-as, de zogenaamde *horizontale verschuivingen*. De DTW-afstandsmeting kan daarentegen wel omgaan met dergelijke verschuivingen. We illustreren dit m.b.v. figuur 12.2. Hier zien we hoe twee sequenties vergeleken worden enerzijds m.b.v. de Euclidische afstandsmeting en anderszijds m.b.v. de DTW-afstandsmeting.

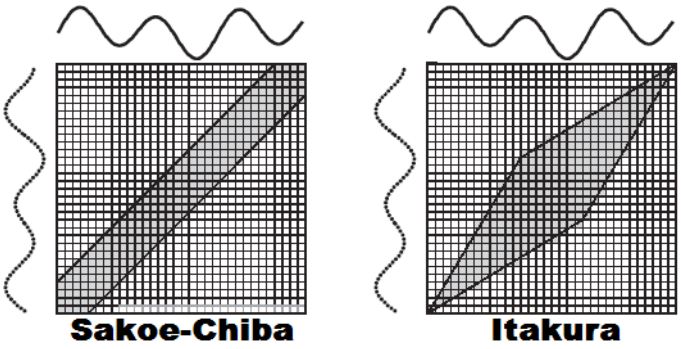


Figuur 12.2: Horizontale verschuiving. ([Keo02])

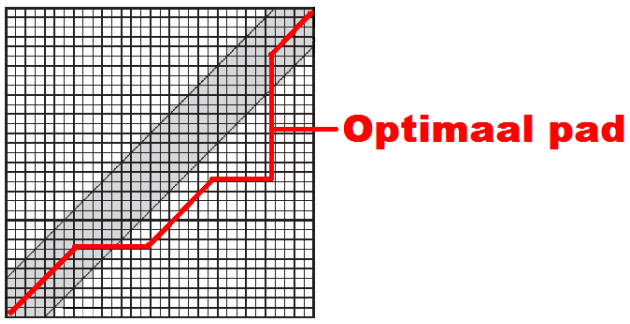
Ten slotte vermelden we dat de Euclidische afstandsmeting als een speciaal geval van de DTW-afstandsmeting kan beschouwd worden. Dit kan echter alleen in het geval dat we twee sequenties van gelijke lengte met elkaar vergelijken. Een pad $W{=}w_1, \ldots, w_K$ beschrijft namelijk de Euclische afstand als er voor elke stap $w_k{=}(i, j)$ (waarbij $1 \leq k \leq K$) steeds geldt dat $k = i = j$.

12.2 Optimalisaties: globale constraints

In [SC90] en [Ita90] worden optimalisaties voor het versnellen van de DTW-afstandsmeting voorgesteld. Beide voorstellen berusten op het idee om niet meer de volledige warping-matrix in rekening te brengen, maar slechts enkel een deel ervan. Dit deel wordt ook wel het *warping-venster* genoemd. Dit venster legt een extra beperking op aan elke stap $w_k=(i,j)$ van het pad $W=w_1,\ldots,w_K$, nl. $j-r \leq i \leq j+r$. Voor het venster voorgesteld in [SC90] heeft r een constante waarde. Naar dergelijk venster wordt ook wel eens verwezen als de *Sakoe-Chiba band*. Bij het *Itakura parallellogram* ([Ita90]) daarentegen neemt r steeds een variabele waarde aan in functie van de waarde van i. Beide warping-vensters worden geïllustreerd m.b.v. figuur 12.3. Het is echter wel mogelijk dat, door te werken met het warping-venster i.p.v. de volledige warping-matrix, het pad met de minste kost in de warping-matrix niet meer gevonden wordt. Dit komt voor indien dit pad geheel of gedeeltelijk buiten het warping-venster valt. Dergelijke situatie wordt getoond in figuur 12.4.



Figuur 12.3: Warping-vensters. ([Keo02])



Figuur 12.4: Optimaal pad dat buiten een warping-venster valt.

Hoofdstuk 13

Similarity search op basis van de DTW-afstand

Nu het begrip DTW-afstand gedefinieerd is, gaan we over tot de behandeling van similarity search op basis van deze afstand. Net als bij similarity search op basis van de Euclidische afstand willen we ook hier weer gebruik maken van een SAM om het proces te versnellen. Hiervoor is het, net als in deel I van dit werk, nodig dat de dimensionaliteit van de sequenties gereduceerd wordt. Daartoe zullen we een techniek beschouwen voorgesteld in [Keo02].

13.1 Reduceren van de dimensionaliteit van sequenties

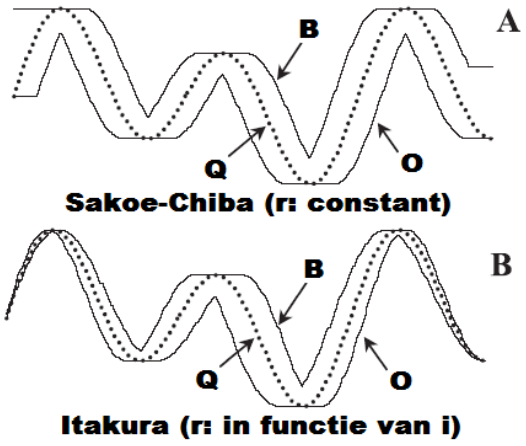
Met het oog op het gebruik van een SAM wordt er, bij het reduceren van de dimensionaliteit van de sequenties, een onderscheid gemaakt tussen querysequenties en datasequenties. De dimensionaliteit van de datasequenties wordt immers gereduceerd aan de hand van een eerder besproken reductietechniek, nl. Piecewise Aggregate Approximation of PAA (zie sectie 8.3). Hierbij wordt een n-dimensionale datasequentie S omgezet naar een N-dimensionaal punt.

Voor de querysequentie hanteren we echter een andere reductietechniek. Bij deze techniek wordt er gebruik gemaakt van de globale constraints van een warping-pad uit sectie 12.2. Met behulp van deze constraints is het namelijk mogelijk om een sequentie langs onder en boven te begrenzen. Zoals gezien drukken dergelijke constraints uit dat voor elke stap w_k=(i,j) van het warping-pad geldt dat $j - r \leq i \leq j + r$. De ondergrens $O(=O_1, \ldots, O_n)$ en bovengrens $B(=B_1, \ldots, B_n)$ worden uitgedrukt in termen van r en Q. Dergelijke grenzen van een sequentie $Q(= q_1, \ldots, q_n)$ worden beschreven door $O_i = \min(q_{i-r}, \ldots, q_{i+r})$ en $B_i = \max(q_{i-r}, \ldots, q_{i+r})$ waarbij $1 \leq i \leq n$.

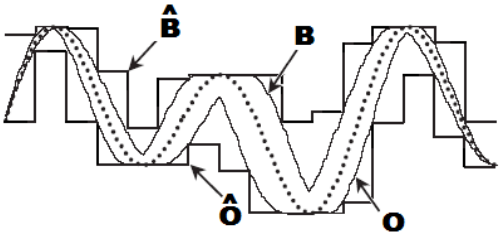
Elke waarde q_i van Q wordt nu duidelijk begrensd door O_i en B_i, m.a.w. $O_i \leq q_i \leq B_i$. Figuur 13.1 illustreert deze onder- en bovengrens. We zien hier hoe deze grenzen opgebouwd zijn zowel in het geval van de Sakoe-Chiba band (r constant) als in het geval van het Itakura parallellogram (r in functie van i). Hiermee wordt de dimensionaliteit van een sequentie Q duidelijk niet gereduceerd. De n-dimensionale sequentie Q wordt hiermee namelijk benaderd m.b.v. twee n-dimensionale punten, nl. de grenzen O en B. Om de dimensionaliteit van deze grenzen te reduceren, maken we gebruik van een PAA-achtige techniek. De twee n-dimensionale punten, O en B, worden omgezet naar twee N-dimensionale punten, $\hat{O}(=\hat{O}_1,\ldots,\hat{O}_N)$ en $\hat{B}(=\hat{B}_1,\ldots,\hat{B}_N)$, m.b.v. de volgende formules

$$\hat{O}_i = \min(O_{\frac{n}{N}(i-1)+1}, \ldots, O_{\frac{n}{N}(i)})$$
$$\hat{B}_i = \max(B_{\frac{n}{N}(i-1)+1}, \ldots, B_{\frac{n}{N}(i)})$$

waarbij $1 \leq i \leq N$. Figuur 13.2 illustreert de nieuwe (gereduceerde) grenzen $\hat{O}$ en $\hat{B}$ van een sequentie.



Figuur 13.1: Onder- en bovengrens van een sequentie. ([Keo02])

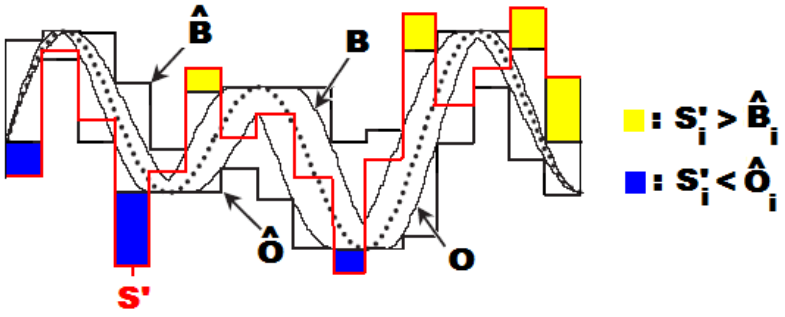


Figuur 13.2: Gereduceerde onder- en bovengrens van een sequentie. ([Keo02])

Zij $S'(= S'_1, \ldots, S'_N)$ het N-dimensionaal punt bekomen uit de datasequentie $S(= S_1, \ldots, S_n)$ d.m.v. van de PAA-reductietechniek en zij $\hat{O}$ en $\hat{B}$ de gereduceerde onder- en bovengrens van een querysequentie Q, dan kunnen we de afstand tussen S' en deze grenzen als volgt berekenen

$$D_{grenzen}(S',\hat{O},\hat{B}) = \sqrt{\sum_{i=1}^{N} \frac{n}{N} \begin{cases} \left(S'_i - \hat{B}_i\right)^2 & S'_i > \hat{B}_i \\ \left(S'_i - \hat{O}_i\right)^2 & S'_i < \hat{O}_i \\ 0 & \hat{O}_i \leq S'_i \leq \hat{B}_i \end{cases}}$$

Figuur 13.3 dient ter verduidelijking van deze $D_{grenzen}()$-meting. In [Keo02] wordt het bewijs geleverd dat er onder dergelijke omstandigheden steeds geldt dat $D_{grenzen}(S',\hat{O},\hat{B}) \leq \text{DTW}(S,Q)$. Een voordeel van de $D_{grenzen}()$-meting is dat deze meting slechts $O(N)$-tijd vereist in tegenstelling tot de $O(n^2)$-tijd voor de DTW-afstandsfucntie.



Figuur 13.3: De $D_{grenzen}()$-afstandsmeting. ([Keo02])

13.2 Subsequence matching

Vooreerst zullen we een methode behandelen voor het probleem van whole matching. Nadien bekijken we hoe we deze methode kunnen uitbreiden met behulp van sliding window technieken om het probleem van subsequence matching aan te pakken. De whole matching methode wordt omschreven m.b.v. de volgende stappen:

- **Initialisatiestap:** Deze stap dient enkel uitgevoerd te worden bij het initialiseren van de database of bij het invoegen van nieuwe datasequenties. Elke n-dimensionale datasequentie S_i wordt hier m.b.v. de PAA-reductietechniek omgezet naar een punt S'_i in de N-dimensionale ruimte. Al deze punten worden vervolgens aan een SAM toegevoegd. Hier

werken we ook weer in het specifieke geval van een R-Tree als de te gebruiken SAM.

- **Stap 1:** De n-dimensionale querysequentie Q wordt, volgens de methode besproken in de voorgaande sectie, omgezet naar twee N-dimensionale punten $\hat{O}$ en $\hat{B}$. Deze punten representeren respectievelijk de (gereduceerde) onder- en de bovengrens van Q (zie figuur 13.2).

- **Stap 2:** Door gebruik te maken van de R-Tree worden alle punten S'_i teruggegeven waarvoor geldt: $D_{grenzen}(S'_i,\hat{O},\hat{B}) \leq \epsilon$. In sectie 5.2.2 hebben we gezien hoe we dergelijke punten met m.b.v. een range search methode kunnen terugvinden in de R-Tree. Deze methode dient echter een kleine aanpassing te ondergaan. Dit komt omdat de querysequentie nu gerepresenteerd wordt aan de hand van twee N-dimensionale punten ($\hat{O}$ en $\hat{B}$) i.p.v. slechts één zoals we eerder hebben gezien. Deze aanpassing zullen we uitvoerig bespreken in sectie 13.3.

- **Stap 3:** Voor elk gevonden punt S'_i grijpen we terug naar de overeenkomstige originele datasequentie S_i. Vervolgens elimineren we alle false alarms door alleen die datasequenties S_i te aanvaarden waarvoor geldt: $DTW(S_i,Q) \leq \epsilon$.

Vermits er voor elke datasequentie S_i geldt dat $D_{grenzen}(S'_i,\hat{O},\hat{B}) \leq \text{DTW}(S_i,Q)$, is het duidelijk dat er zich bij deze methode geen false dismissals zullen voordoen. M.a.w. zal deze methode dus alle datasequenties S_i teruggeven waarvoor geldt $DTW(S_i,Q) \leq \epsilon$.

Zoals reeds vermeld, is het mogelijk de voorgaande methode m.b.v. een sliding window techniek uit te breiden tot een subsequence matching methode. De volgende stappen beschrijven deze resulterende methode:

- **Initialisatiestap:** Deze stap dient ook hier weer enkel uitgevoerd te worden bij het initialiseren van de database of bij het invoegen van nieuwe datasequenties. Elke datasequentie S_i wordt m.b.v. een sliding window techniek opgesplitst in windows van lengte ω. Vervolgens past men de PAA-reductietechniek toe op elk van de bekomen windows. Elke n-dimensionale window w_k wordt m.a.w. gereduceerd naar een N-dimensionaal punt. Al deze punten worden vervolgens aan een R-Tree toegevoegd.

- **Stap 1:** De querysequentie Q wordt op haar beurt in windows van lengte ω opgesplitst. Iedere bekomen window w_j wordt omgezet naar twee N-dimensionale punten $\hat{O}_j$ en $\hat{B}_j$ volgens de methode besproken in de voorgaande sectie, waarbij ($1 \leq j \leq$ *totaal aantal querywindows*).

Hierbij staat het punt $\hat{O}_j$ voor de (gereduceerde) ondergrens van window w_j. Het punt $\hat{B}_j$ staat dan ook logischerwijs voor de (gereduceerde) bovengrens van de desbetreffende window.

- **Stap 2:** Op analoge wijze als in stap twee van de voorgaande whole matching methode wordt elk koppel $(\hat{O}_j,\hat{B}_j)$ vergeleken met al de punten in de SAM. De resulterende verzameling van punten vormt de kandidatenverzameling, zoals beschreven in hoofdstuk 4.

- **Stap 3:** Voor ieder punt in de kandidatenverzameling grijpen we terug naar zijn overeenkomstige datasequentie S_i. Door voor elk punt bij te houden met welk koppel $(\hat{O}_j,\hat{B}_j)$ het juist overeenkomt, kunnen we een bepaalde regio S_{regio} van lengte $|Q|$ in de overeenkomstige datasequentie S afbakenen. Voor elke S_{regio} wordt er vervolgens gecontroleerd of deze deelsequentie voldoet aan: $DTW(S_{regio},Q) \leq \epsilon$. Op deze wijze blijven alleen de (subsequence) matchings over en worden alle eventuele false alarms uit de kandidatenverzameling verwijderd.

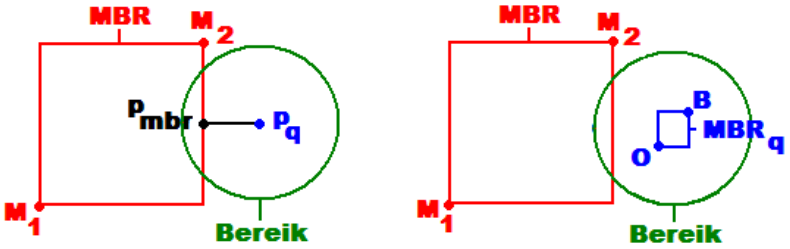
In bovenstaande stappen kunnen we duidelijk het meerstappenproces uit sectie 4.1 in herkennen.

13.3 Aanpassing range search

In sectie 5.2.2 hebben we gezien hoe we alle punten in de R-Tree m.b.v. een range search methode kunnen vinden die zich binnen een bereik ϵ van een opgegeven punt p_q bevinden. Bij de similarity search methoden uit de voorgaande sectie hebben we gemerkt dat we nu echter te maken hebben met twee punten (i.p.v. p_q), nl. $\hat{O}$ en $\hat{B}$. We bekijken nu hoe we de range search methode hiervoor zullen aanpassen.

In plaats van te werken met het punt p_q, werken we nu met een MBR M_q. Vermits elke MBR beschreven wordt aan de hand van twee punten M_1 en M_2, gebruiken we voor de beschrijving van M_q de punten $\hat{O}$ en $\hat{B}$. Dit betekent m.a.w. dat M_1=$\hat{O}$ en M_2=$\hat{B}$. Voor de verdere behandeling zullen we dan ook steeds $\hat{O}$ en $\hat{B}$ gebruiken als beschrijvingspunten voor M_q. Voor de andere MBR's gebruiken we steeds M_1 en M_2. Bij de range search methode uit sectie 5.2.2 komt het er dus op neer om de MBR's uit de R-Tree steeds te vergelijken met het punt p_q. In deze aangepaste range search methode vergelijken we deze MBR's daarentegen met M_q. Dit verschil wordt geïllustreerd in figuur 13.4.

Opnieuw wordt er een onderscheid gemaakt tussen de MBR's in de R-Tree, nl. de MBR's in de bladeren en deze in de interne knopen. Zoals



Figuur 13.4: Aanpassing range search methode

ondertussen meermaals vermeld, stellen de MBR's in de bladeren niets meer voor dan het punt dat ze representeren. Voor dergelijke MBR's geldt dus $M_1 = M_2(= m_1, \ldots, m_N)$. Om deze MBR's te vergelijken met M_q maken we gebruik van de $D_{grenzen}()$-afstandsmeting en dit gebeurt als volgt

$$D_{grenzen}(M1,\hat{O},\hat{B}) = \sqrt{\sum_{i=1}^{N} \frac{n}{N} \begin{cases} \left(m_i - \hat{B}_i\right)^2 & m_i > \hat{B}_i \\ \left(m_i - \hat{O}_i\right)^2 & m_i < \hat{O}_i \\ 0 & \hat{O}_i \leq S'_i \leq \hat{B}_i \end{cases}}$$

Voor het vergelijken van M_q met de MBR's in de interne knopen maken we echter gebruik van een aangepaste $D_{grenzen}()$-afstandsmeting. Zij M_1 $(= min_1, \ldots, min_N)$ en M_2 $(= max_1, \ldots, max_N)$ de punten die een MBR in een interne knoop beschrijven dan komt dit neer op

$$D_{aangepast}(M1,M2,\hat{O},\hat{B}) = \sqrt{\sum_{i=1}^{N} \frac{n}{N} \begin{cases} \left(min_i - \hat{B}_i\right)^2 & min_i > \hat{B}_i \\ \left(max_i - \hat{O}_i\right)^2 & max_i < \hat{O}_i \\ 0 & \text{andere gevallen} \end{cases}}$$

Nu we weten hoe we M_q met de MBR's in de R-Tree kunnen vergelijken behandelen we de aangepaste range search methode. Deze methode wordt beschreven aan de hand van pseudocode 5. We beginnen hierbij nog steeds in de top van de boom en we werken nog steeds in de richting van de bladeren. Ook het principe van het snoeien van de boom blijft behouden. Het enige dat verandert, is dat we nu de MBR's in de R-Tree vergelijken met de MBR M_q en niet meer met het punt p_q.

Pseudocode 5 ZoekPunten2(MBR M_q,KNOOP *huidigeKnoop*,BEREIK ϵ)

```
/* M_q wordt beschreven door de punten B^ en O^. */
if (huidigeKnoop is een blad) then
  /* We bevinden ons in een blad dus alle MBR's worden beschreven
     door twee punten M_1 en M_2, waarbij M_1=M_2. */
  for (alle tuples (id_i,mbr_i) van huidigeKnoop, i = 1,...,#tuples) do
    if (D_grenzen(M_1,B^,O^)≤ ε) then
      resultaat += id_i;
    end if
  end for
else if (huidigeKnoop is een interne knoop) then
  /* We bevinden ons in een interne knoop dus alle MBR's worden
     beschreven door twee verschillende punten M_1 en M_2. */
  for (alle tuples (ptr_i,mbr_i) van huidigeKnoop, i = 1,...,#tuples) do
    if (D_aangepast(M_1,M_2,B^,O^)≤ ε) then
      resultaat += ZoekPunten2(M_q,GeefKnoop(ptr_i),ε);
    end if
  end for
end if
return resultaat;
```

Hoofdstuk 14

Testresultaten: Dynamic Timewarping

In dit hoofdstuk bespreken we de testresultaten van de geïmplementeerde technieken met betrekking tot similarity search op basis van de DTW-afstand. Ook hier beschouwen we eerst een sequentiële scan methode. Net als bij de testen van de voorgaande technieken werken we ook hier weer met een tolerantie $\epsilon = 0,5$. Dit stelt ons namelijk gemakkelijk in staat relevante vergelijkingen te maken.

14.1 Sequentiële scan

Het vergelijken ($\epsilon = 0,5$) van een querysequentie van lengte 128 met de 10144 datasequenties in de database m.b.v. een sequentiële scan vergt rond de drie uur en een kwartier. In vergelijking met de sequentiële scan op basis van de Euclidische afstandsmeting is dit een enorm tijdsverschil. Dit is natuurlijk te wijten aan het feit dat de Euclidische afstandsmeting slechts $O(n)$-tijd vergt tegenover de $O(n^2)$-tijd van de DTW-afstandsmeting, waarbij n de lengte van de sequenties voorstelt.

Tevens hebben we deze sequentiële scan meermaals herhaald, maar dan door de globale constraints van de DTW-afstandsmeting in rekening te brengen. De testresultaten van de 'Sakoe-Chiba band'-constraint enerzijds en van de 'Itakura parallellogram'-constraint anderzijds bleken niet erg van elkaar te verschillen. We zullen hier dan ook niet dieper ingaan op het verschil tussen beide constraints, maar we zullen de globale constraints meer algemeen bekijken. We hebben deze constraints namelijk getest voor $r = 32$ en $r = 50$ en vergeleken met de gewone (zonder constraints) DTW-afstandsmeting. Dit leverde ons voor $r = 32$ en $r = 50$ respectievelijk een tijdswinst van gemid-

deld 42 en 28 minuten op. Voor $r = 32$ gaf de DTW-afstandsmeting in 33% van zijn metingen een andere waarde dan de DTW-afstandsmeting zonder globale constraint. Voor $r = 50$ gaf dit slechts voor 4% van de totale metingen een verschil. Gezien de verhouding tijdswinst versus precisie lijkt het gebruik van de globale constraint met $r = 50$ zeker de moeite waard.

14.2 Dynamic Timewarping met index

De testresultaten van deze laatste methode zijn lang niet zo spectaculair als voorheen bij de GEMINI-methode. Het vergelijken van een querysequentie van lengte 128 met de 10144 datasequenties, waarbij $\epsilon = 0,5$ en $r = 50$, vergt immers nog steeds om een bij de twee uur tijd. Dit is enerzijds te wijten aan het feit dat er bij similarity search op basis van de DTW-afstand veel meer matchings gevonden worden dan op basis van de Euclidische afstandsmeting (met dezelfde tolerantie). Door de waarde van ϵ bij onze tests te verlagen, daalt de benodigde tijd echter nog steeds niet spectaculair. We vermoeden dat dit komt doordat deze methode, zoals gezien in vorig hoofdstuk, werkt met een gereduceerde onder- en bovengrens voor de querysequenties. Deze grenzen beschrijven deze sequentie vaak veel ruimer dan bijvoorbeeld de reductietechnieken dat doen bij de GEMINI-methode. Deze technieken benaderen de orginele querysequentie m.a.w. veel beter dan deze onder- en bovengrens. Dit brengt uiteraard veel meer false alarms met zich mee, wat uiteraard een negatieve uitwerking heeft op de zoektijd. Ons vermoeden wordt gesteund door het feit dat de inhoud van de kandidatenverzameling soms tot meer dan 90% bestond uit false alarms. Het gebruik van de R-Tree levert wel een positieve bijdrage, want er worden immers toch een aantal takken gesnoeid. Deze bijdrage is echter wel niet van de aard zoals bij de GEMINI-methode.

Hoofdstuk 15

Toekomstige voortzetting

In de werkstukken [Bol05] en [Hin05] werd een dynamisch databasesysteem ontwikkeld ten bate van Medtronic BRC. Hierbij werd een dynamische structuur uitgewerkt om informatie over patienten te bewaren en indien nodig op te vragen. Tevens werd veel aandacht besteed aan de wijze waarop de ECG's van patienten in de database beheerd kunnen worden. We zouden dit werk kunnen beschouwen als een verlengstuk van deze werkstukken. We hebben in dit werk voornamelijk onderzoek verricht naar het aspect van similarity search van de ECG-sequenties aanwezig in de database. Dit heeft echter nog niet geleid tot een bruikbare applicatie. Het opzet van dit hoofdstuk is om aan te geven welke stappen in de toekomst daartoe ondernomen kunnen worden.

15.1 Normaliseren van de ECG-sequenties

Zoals vermeld hebben we in de werk gewerkt met sequenties uit de MIT-BIH Arrhythmia Database. Deze sequenties zijn gekenmerkt door enkele specifieke eigenschappen (zie sectie 2.3). Een zeer belangrijke eigenschap hiervan is dat alle samples van deze sequentie over een waarde tussen de -5 mV en de $+5$ mV beschikken. Deze samples zijn dus allemaal gelegen binnen het interval $[-5\,mV, +5\,mV]$. Een probleem dat zich kan voordoen wanneer we andere sequenties bij ons werk willen gaan betrekken, is dat de waarden van de samples van deze nieuwe sequenties mogelijk binnen een ander interval liggen. Hiervoor raden we dan ook aan om deze sequenties, alvorens ze aan de database toe te voegen, te normaliseren in de richting van de Y-as. Op deze wijze bekomen we dat de waarden van alle samples van elke sequentie in de database binnen hetzelfde interval (b.v. $[-5\,mV, +5\,mV]$) liggen. Hiervoor werd echter in dit werk nog niets voorzien.

Een andere invloedrijke eigenschap van de ECG-sequenties uit de MIT-BIH Arrhythmia Database is dat deze sequenties aan 360 Hz gedigitaliseerd zijn. Om deze sequenties te kunnen vergelijken met sequenties die aan een andere Hz gedigitaliseerd zijn, is het nodig dat ook hiervoor een oplossing gevonden wordt. Hierbij denken we weer aan het op voorhand normaliseren van de sequenties. Een andere oplossing zou kunnen zijn om de gebruikte similarity search methode uit te breiden met scaling-technieken. De query-sequentie wordt dan ook nog eens onder verschillende schalen vergeleken met de datasequenties. Uiteraard zal deze laatste oplossing het zoekproces merkelijk vertragen.

15.2 Implementatie R-Tree

Vermits de werkstukken van [Bol05] en [Hin05] gebouwd zijn rond een Oracle-database, hebben we getracht om ons in dit werk daar zo goed mogelijk aan te houden. Voor het gebruik van een R-Tree als SAM hebben we ons dan ook eerst gericht naar Oracle Spatial ([Ora]). Geometrische objecten (punten, lijnstukken, ...) met een dimensionaliteit hoger dan vier worden in deze implementatie van de R-Tree echter niet ondersteund. Een tweede probleem hierbij is dat de meeste functies, speciaal voorzien voor deze R-Tree-implementatie, enkel en alleen te gebruiken zijn indien er zich 2-dimensionale objecten in de R-Tree bevinden. Om deze redenen hebben we voor het testwerk van deze thesis beroep gedaan op een andere implementatie van de R-Tree, nl. deze uit [Had]. Het probleem dat zich hier momenteel mee voordoet is dat deze relatief los staat van de Oracle database. Voor een verdere voortzetting van dit werk zal ook dit euvel uit de weg geruimd moeten worden.

15.3 Nieuwe reductietechnieken

In deze thesis hebben we reeds enkele technieken voor het reduceren van de dimensionaliteit van de sequenties besproken. Een mogelijke verderzetting van dit werk zou zijn om enkele andere reductietechnieken te bestuderen en deze eventueel toe te passen. De mogelijkheden hierbij zijn enorm. Zo zouden we kunnen proberen andere discrete wavelet transformaties te gebruiken, zoals b.v. de Daub4 tranformaties ([Dau92]). Hierbij kan er eventueel ook gezocht worden naar specifieke transformaties waarmee de ECG-sequenties goed beschreven kunnen worden. We moeten wel blijven toezien dat een eventuele nieuwe reductietechniek geen false dismissals met zich mee zou

brengen.

15.4 Input- en outputfaciliteiten

Het inbrengen van de behandelde similarity search methoden in de applicatie, ontwikkeld in [Bol05] en [Hin05], is een volgende stap die dient verwezenlijkt te worden. Hierbij is het belangrijk dat er een overzichtelijke GUI voorzien wordt om deze methoden te kunnen gebruiken. Zo moet er bijvoorbeeld een eenvoudige inputfaciliteit voorzien worden om de querysequenties te specifiëren. Dit zou kunnen verwezenlijkt worden door een aantal standaardsequenties beschikbaar te stellen. Hieruit zou men dan steeds een keuze kunnen maken om ze te gebruiken als querysequentie. Het queriën blijft dan echter wel beperkt tot het zoeken met de standaardsequenties. Een eventuele oplossing hiervoor zou kunnen zijn, nl. de mogelijkheid te voorzien dat een querysequentie kan geupload worden. Op deze wijze zou men niet beperkt zijn tot alleen de standaardsequenties. Het is duidelijk dat op dit vlak nog enig overleg noodzakelijk is. Tevens zal het nodig zijn dat resultaten van een query overzichtelijk weergegeven worden. Hiervoor zullen ook de nodige outputfaciliteiten moeten voorzien worden.

15.5 Afspraken en keuzes

Tot slot vermelden we dat voor de mogelijke voortzetting van dit werk de nodige afspraken en keuzes dienen gemaakt te worden. Ten eerste zal er een keuze moeten gemaakt worden over welke aspecten er bij het queriën van belang zijn. Een voorbeeld hiervan is of we bij het vergelijken van sequenties al dan niet rekening houden met horizontale of verticale verschuivingen. Deze keuzes hebben immers een grote invloed op welke similarity search methode we moeten gebruiken.

Een tweede zeer belangrijke afspraak die dient gemaakt te worden betreft de vorm van de querysequenties. Er zal minstens een akkoord moeten bereikt worden over de lengte van deze querysequenties. We hebben immers gemerkt dat de lengte van de querysequenties een zeer belangrijke rol speelt bij de sliding window technieken. Hoe groter de beoogde querysequenties, hoe groter de windows die we kunnen gebruiken. Grotere windows leiden, zoals gezien, tot minder false alarms en dus tot snellere query's. Vermits een querysequentie nooit korter mag zijn dan de afgesproken windowlengte, is het nadeel hiervan dat men zich, na het maken van deze keuze, strak zal moeten houden aan de gekozen querylengte.

Hoofdstuk 16

Conclusie

In de eerste inleidende hoofdstukken hebben we getracht een overzicht te geven van de benodigde basiskennis over similarity search van tijdreeksen. Hierbij werden de belangrijkste basisbegrippen omtrent dit onderwerp strak gedefinieerd. Tevens werden er enkele sliding window technieken behandeld. Deze technieken maken het mogelijk een whole matching methode uit te breiden tot een methode voor het aanpakken van het subsequence matching probleem. Vermits we in dit werk voornamelijk het subsequence matching probleem aangepakt hebben, werden deze technieken dan ook grondig behandeld. Hierbij is gebleken dat de General Match methode enorm interessant is, vermits het de voordelen van zowel de FRM methode als de Dual Match methode weet te combineren.

Tevens is ons duidelijk geworden dat elke vorm van similarity search van tijdreeksen in essentie gebaseerd is op een afstandsmeting. In deze thesis hebben we op basis hiervan een opdeling gemaakt in twee categorieën, nl. similarity search op basis van de Euclidische metriek en similarity search op basis van de dynamic timewarping afstand. Deze categorieën worden respectievelijk in deel I en deel II van dit werk behandeld.

Deel I werd aangevat met de behandeling van de GEMINI-methode. Hierbij wordt er beroep gedaan op een spatial access methode (SAM) en technieken voor het reduceren van de dimensionaliteit van tijdreeksen. Voor het testwerk in deze thesis hebben we ons specifiek toegelegd op tijdreeksen afkomstig van ECG-sequenties. Ook hebben we steeds specifiek de R-Tree gebruikt als SAM. Uit de testresultaten is gebleken dat het gebruik van een R-Tree een zeer grote invloed kan hebben op de zoektijden. Tevens lijkt de PAA-techniek volgens onze testresultaten de ECG-sequenties het best te beschrijven, net iets beter dan de DFT en de DWT. De GEMINI-methode blijkt ook het snelst te werken van al de methoden die we behandeld hebben, maar vindt echter wel het kleinst aantal subsequence matchings. Dit

komt o.a. omdat deze methode niet kan inspelen op horizontale verschuivingen. Desondanks lijkt deze methode het meest aangewezen om een bruikbare applicatie omtrent similarity search van ECG-sequenties te ontwikkelen.

Een tweede methode die aan bod kwam in deel I van dit werk is de minimale variantie matching methode. Deze methode heeft als voordeel dat het de querysequenties kan uitrekken over de gehele lengte van een datasequentie. Zo kan deze methode, in tegenstelling tot de GEMINI-methode, wel inspelen op horizontale verschuivingen. Een nadeel van deze methode is dat er hier geen SAM aan te pas komt. Dit resulteert, zoals in onze testresultaten te bemerken viel, in zeer lange zoektijden. Deze methode lijkt ons dan ook zeker niet aangewezen om te gebruiken met betrekking tot de vrij lange ECG-sequenties.

In ons tweede en laatste deel komt dan similarity search op basis van de dynamic timewarping afstand aan bod. Hierbij behandelen we een methode waarbij er, net als bij de GEMINI-methode, beroep wordt gedaan op een SAM om de zoektijden te verbeteren. Ook deze methode biedt een oplossing voor de eventuele horizontale verschuivingen. Uit onze testresultaten is echter gebleken dat ook deze methode vrij lange zoektijden met zich meebrengt. Het effect van de SAM, in ons geval de R-Tree, blijkt bijlange niet zo spectaculair te zijn als bij de GEMINI-methode. Dit is vermoedelijk te wijten aan het feit dat de querysequenties te ruw benaderd worden m.b.v. een onder- en een bovengrens. Hieruit concluderen we dat deze techniek met het oog op een bruikbare applicatie waarschijnlijk de duimen moet leggen voor de GEMINI-methode.

Tot slot kunnen we besluiten dat er nog enkele problemen overwonnen dienen te worden, alvorens dit alles tot een efficiënt bruikbare applicatie kan leiden. Rome werd echter ook niet op één dag gebouwd.

Bibliografie

[AFS93] Rakesh Agrawal, Christos Faloutsos, and Arun N. Swami. Efficient similarity search in sequence databases. In *Proceedings of the 4th International Conference of Foundations of Data Organization and Algorithms*, pages 69–84. Springer Verlag, 1993.

[Bol05] Jan Bollen. *Binary data management in databases: Theoretical study and application in a database of heart signals. Master thesis.* University of Limburg (LUC), Department WNI, 2004-2005.

[CF99] Kin-Pong Chan and Ada Wai-Chee Fu. Efficient time series matching by wavelets. In *ICDE*, pages 126–133, 1999.

[CFY03] Kin-Pong Chan, Whai-Chee Fu, and Clement Yu. Haar wavelets for efficient similarity search of time-series: with and without time warping. *IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 3*, pages 686–705, 2003.

[Chu92] Charles K. Chui. *An introduction to wavelets.* Academic Press Professional, Inc., San Diego, CA, USA, 1992.

[Dau92] Ingrid Daubechies. *Ten lectures on wavelets.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1992.

[DL42] G. C. Danielson and C. Lanczos. Some improvements in practical fourier analysis and their application to x-ray scattering from liquids. *J. Franklin Inst. 233*, pages 365–380, 1942.

[Fal96] Christos Faloutsos. *Searching Multimedia Databases by Content.* Kluwer Academic Publishers, 1996.

[FRM94] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings ACM SIGMOD Conference, Mineapolis, MN*, pages 419–429, 1994.

[Gut84] Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *ACM SIGMOD*, pages 47–57, 1984.

[Had] Marios Hadjieleftheriou. *Rtree: implementation*. National Technical University of Athens Knowledge and Database Systems Laboratory.
http://www.dbnet.ece.ntua.gr/ mario/rtree/.

[Hin05] Ben Hinssen. *Attribute management in databases: Theoretical study and application in a database of heart signals. Master thesis*. University of Limburg (LUC), Department WNI, 2004-2005.

[HS03] Gisli R. Hjaltason and Hanan Samet. Index-driven similarity search in metric spaces. *ACM Trans. Database Syst.*, 28(4):517–580, 2003.

[Ita90] Fumitada Itakura. Minimum prediction residual principle applied to speech recognition. pages 154–158, 1990.

[KCPM01] Eamonn J. Keogh, Kaushik Chakrabarti, Michael J. Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, 3(3):263–286, 2001.

[Keo02] Eamonn J. Keogh. Exact indexing of dynamic time warping. In *VLDB*, pages 406–417, 2002.

[KP99] Eamonn Keogh and M. Pazzani. Scaling up dynamic time warping to massive datasets. In J. M. Zytkow and J. Rauch, editors, *3rd European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'99)*, volume 1704, pages 1–11, Prague, Czech Republic, 1999. Springer.

[LMW+05a] L. Latecki, V. Megalooikonomou, Q. Wang, R. Lakaemper, C. Ratanamahatana, and E. Keogh. Elastic partial matching of time series, 2005.

[LMW+05b] Longin Jan Latecki, Vasileios Megalooikonomou, Qiang Wang, Rolf Lakaemper, C. A. Ratanamahatana, and E. Keogh. Partial elastic matching of time series. *icdm*, 0:701–704, 2005.

[MWH02] Yang-Sae Moon, Kyu-Young Whang, and Wook-Shin Han. Generalmatch: A subsequence matching method in time-series

databases based on generalized windows. In *SIGMOD Conference*, pages 382–393, 2002.

[MWL01a] Yang-Sae Moon, Kyu-Young Whang, and Woong-Kee Loh. Efficient time-series subsequence matching using duality in constructing windows. *Information Systems*, 26(4):279–293, 2001.

[MWL01b] Yang-Sea Moon, Kyu-Young Whang, and Woong-Kee Loh. Duality-based subsequence matching in time-series databases. In *Proc. IEEE Internat. Conf. on Data Engineering ICDE*, pages 263–272, 2001.

[Nat] National Institute of Standards and Technology. http://www.nist.gov/.

[Ora] Oracle. *Oracle Spatial Documentation.* http://www.oracle.com/technology/documentation/spatial.html.

[PBFV88] W.H. Press, S.A. Teukolsky B.P. Flannery, and W.T. Vetterling. *Numerical Recipes in C.* Cambridge, 1988.

[Res] Research Resource for Complex Physiologic Signals. *MIT-BIH Arrhythmia Database.* http://www.physionet.org/physiobank/database/mitdb/.

[RJ93] Lawrence Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

[SC90] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. pages 159–165, 1990.

[UZ] UZ Leuven. *Bouw van het normale hart.* http://www.uzleuven.be/kindercardiologie/normaalhart.

[WAA00] Yi-Leh Wu, Divyakant Agrawal, and Amr El Abbadi. A comparison of DFT and DWT based similarity search in time-series databases. In *CIKM*, pages 488–495, 2000.

[Yan] Frank G. Yanowitz. *ECG Learning Center.* University of Utah School of Medicine. http://www-medlib.med.utah.edu/kw/ecg/index.html.

[YJF98] Byoung-Kee Yi, H. V. Jagadish, and Christos Faloutsos. Efficient retrieval of similar time sequences under time warping. In *ICDE*, pages 201–208, 1998.

Auteursrechterlijke overeenkomst

Opdat de Universiteit Hasselt uw eindverhandeling wereldwijd kan reproduceren, vertalen en distribueren is uw akkoord voor deze overeenkomst noodzakelijk. Gelieve de tijd te nemen om deze overeenkomst door te nemen en uw akkoord te verlenen.

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Database search van tijdreeksen, met toepassing in de firma Medtronic
Richting: **Master in de informatica** Jaar: **2006**
in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Deze toekenning van het auteursrecht aan de Universiteit Hasselt houdt in dat ik/wij als auteur de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij kan reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

U bevestigt dat de eindverhandeling uw origineel werk is, en dat u het recht heeft om de rechten te verlenen die in deze overeenkomst worden beschreven. U verklaart tevens dat de eindverhandeling, naar uw weten, het auteursrecht van anderen niet overtreedt.

U verklaart tevens dat u voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen hebt verkregen zodat u deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal u als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze licentie

Ik ga akkoord,

Tom KELLENS

Datum: