Het gebruik van memetische algoritmen bij operationele beslissingen

Een toepassing op het vestigingsprobleem

Stéphanie JANSEN

promotor :
Prof. dr. Gerrit JANSSENS

Eindverhandeling voorgedragen tot het bekomen van de graad
Handelsingenieur major operationeel management en logistiek

universiteit hasselt

Woord vooraf

Deze eindverhandeling vormt het sluitstuk van mijn opleiding als Handelsingenieur aan de Universiteit Hasselt te Diepenbeek. De realisatie hiervan zou onmogelijk geweest zijn zonder de medewerking, hulp en steun van een aantal personen en daarom zijn enkele oprechte woorden van dank hier zeker niet misplaatst.

Op de eerste plaats wil ik mijn promotor Prof. dr. Gerrit Janssens bedanken. Zonder hem had ik mijn eindverhandeling niet tot een goed einde kunnen brengen. Graag wil ik ook dr. Kenneth Sörensen bedanken voor zijn vrijwillige hulp en tijd die hij gestoken heeft in het ontwerpen van het programma voor mijn onderzoek. Hun kennis, deskundig advies en opbouwende kritiek waren onontbeerlijk voor het uitvoeren van dit onderzoek.

Tot slot wil ik mijn ouders en mijn vrienden bedanken voor de steun die ik dit jaar van hen heb ontvangen. Christophe, Maxime en Martijn wil ik extra bedanken voor hun interesse, morele steun en praktische hulp. Mijn ouders wil ik voornamelijk bedanken voor de financiële steun die ik heb mogen genieten gedurende heel mijn opleiding.

Samenvatting

Veel bedrijven worden al eens geconfronteerd met het bepalen van nieuwe vestigingsplaatsen. Het is een belangrijke lange-termijn beslissing aangezien met het bouwen of openen van vestigingen hoge kosten gepaard gaan. Daarom is het belangrijk om te proberen deze hoge kosten te minimaliseren.

Nadat aan de hand van kwalitatieve determinanten een reeks potentiële vestigingen geselecteerd zijn, dienen de vestigingen geopend te worden die over voldoende capaciteit beschikken om aan de totale vraag van de klanten te voldoen én die de totale kosten proberen te minimaliseren. In deze eindverhandeling wordt gekozen voor het *capacitated fixed charge location* model. Dit model sluit aan bij de vereisten.

Naast een aantal bestaande oplossingsmethoden voor dit model worden de memetische algoritmen ingeleid. Het memetisch algoritme dat in het onderzoek gebruikt wordt, wordt weergegeven en duidelijk toegelicht. De drie parameters in het algoritme, namelijk de populatiegrootte, het aantal generaties en de kans op *local search* kunnen in het onderzoek elk twee waarden aannemen. Zo bekomt men acht verschillende parametercombinaties voor het memetisch algoritme.

Om verschillende invalshoeken te hebben, zijn datasets opgesteld voor een kleine, middelgrote en grote onderneming. Voor elk van die drie ondernemingen liggen de klanten telkens ofwel verspreid, ofwel geclusterd. Ook wordt gespeeld met de capaciteiten van de potentiële vestigingen. Bij de eerste reeks hebben de potentiële vestigingen elke keer een variërende capaciteit. Bij een tweede reeks beschikken de potentiële vestigingen iedere keer over identieke capaciteiten. Hoofd- en interactie-effecten worden berekend. Hiervan wordt een betrouwbaarheidsinterval opgesteld om te zien of het hoofd- of interactie-effect effectief significant is.

Alle kleine en grote datasets geven iedere keer dezelfde optimale oplossing weer. De hoofd- en interactie-effecten bedragen hier nul. Een wijziging in de parameterwaarden heeft alzo geen invloed op de totale kost. Dit geldt ook voor de middelgrote ondernemingen waar de klanten geclusterd liggen. Indien de klanten van de middelgrote onderneming willekeurig verspreid liggen, geven de acht parametercombinaties verschillende uitkomsten aan. Slechts weinig effecten zijn significant verschillend van nul wanneer we een betrouwbaarheidsinterval van 95% of 99% selecteren.

Zowel bij de dataset waar de potentiële vestigingen variërende capaciteiten hebben als waar de potentiële vestigingen over identieke capaciteiten beschikken heeft het hoofdeffect van de kans op *local search* een significante invloed op het 95%-niveau. Wanneer de kans stijgt tot 80%, leidt dit tot een vermindering in de totale kosten van beide datasets.

Bij de dataset waar de potentiële vestigingen variërende capaciteiten hebben is het hoofdeffect horende bij een stijging van de populatiegrootte ook significant verschillend van nul. Door een stijging van 20 naar 40 individuen, daalt gemiddeld de totale kost. De overige effecten zijn allemaal niet significant verschillend van nul.

Verschillende verklaringen kunnen gegeven worden voor de behaalde resultaten. De verhouding van de totale vraag tot de totale beschikbare capaciteit van een dataset kan van invloed zijn. Een lage verhouding geeft het algoritme meer toelaatbare oplossingen wat leidt tot meer rekenwerk.

Een tweede verklaring voor de resultaten wordt gezocht bij de ligging van de klanten. Wanneer de klanten in clusters liggen, worden de vestigingen geopend die het dichtst bij de clustercentra gelegen zijn. Indien een verder gelegen vestiging zou geopend worden, lopen de transportkosten ineens hoog op. Bij de willekeurig verspreide klanten kan de transportkost geminimaliseerd worden door meer vestigingen te openen.

Inhoudsopgave

1. Probleemstelling en werkwijze

1.1 Praktijkprobleem

In deze eindverhandeling wordt een oplossingsmethode voor het vestigingsprobleem uitgewerkt. Verschillende parameters van het vestigingsprobleem en het memetisch algoritme worden onderscheiden. De redenen voor de keuze van dit onderwerp zullen nu toegelicht worden.

Het vestigen van gebouwen is een belangrijke strategische beslissing. Zowel fabrieken, opslagplaatsen, fastfoodketens als scholen, hospitalen of brandweerwagendepots hechten belang aan het zo optimaal mogelijk plaatsen van hun gebouwen in de omgeving. (Coyle et al., 1992; Hansen 1987)

Vooreerst is het zinvol te weten voor welke omvang van gegevens een oplossingsmethode consequente resultaten kan leveren. Gegevens verzamelen kost tijd en bijgevolg ook geld. Daarom is het handig te weten welke wijzigingen in parameters een bepaalde significante invloed hebben op de kwaliteit van de resultaten die verkregen worden.

De bedoeling van het onderzoek is na te gaan voor een aantal parameters van het memetisch algoritme of zij goede resultaten verschaffen voor de datasets. Zo kan tijd uitgespaard worden door het algoritme in te korten of kan aan kwaliteit gewonnen worden door het algoritme te verlengen.

Verschillende oplossingsmethoden om dit probleem aan te pakken zijn reeds onderzocht. In deze eindverhandeling wordt een relatief recente oplossingsmethode van naderbij bekeken, namelijk memetische algoritmen.

1.2 Centrale onderzoeksvraag

De centrale onderzoeksvraag die uit dit praktijkprobleem kan afgeleid worden, luidt als volgt:

"Wat is het effect op de resultaten van het memetisch algoritme dat wordt toegepast op een vestigingsprobleem indien verscheidene parameters gewijzigd worden?"

1.3 Deelvragen

Vertrekkende van de centrale onderzoeksvraag en het praktijkprobleem kunnen volgende deelvragen geformuleerd worden.

Om de onderzoeksvraag een gegrond antwoord te kunnen geven dient vooreerst het vestigingsprobleem nader bekeken te worden. Om een beter inzicht in het probleem te krijgen, is het nuttig om de veelzijdigheid van dit probleem te benadrukken. Hiertoe kan de volgende deelvraag worden gesteld:

- *In welke klassen kan het vestigingsprobleem opgedeeld worden?*

Nadat een klasse gekozen is dient een model geselecteerd te worden. Het vestigingsprobleem kan, afhankelijk van de doelstelling, geformuleerd worden in een bepaald wiskundig model:

- *Welke zijn de belangrijkste modellen in de gekozen categorie om het vestigingsprobleem op te lossen?*

Aangezien wij op zoek zijn naar de resultaten van een specifieke oplossingsmethode lijkt het aangewezen om na te gaan via welke methodes het vestigingsprobleem reeds is opgelost. Hiertoe wordt de volgende deelvraag voorzien:

- *Via welke methodes kan het vestigingsprobleem worden opgelost?*

Vervolgens kan overgegaan worden tot het onderzoek. Dit gebeurt aan de hand van een memetisch algoritme. Aangezien dit algoritme vele parameters bevat, kunnen volgende deelvragen gesteld worden:

- *Welke parameters zijn belangrijk bij het memetisch algoritme?*
- *Welke parameters spelen een rol bij het vestigingsprobleem?*
- *In welke mate gaan we deze parameters wijzigen?*
- *In hoeverre hebben deze parameters invloed op de resultaten?*

1.4 Onderzoeksopzet

Een aanzienlijk deel van deze eindverhandeling is gebaseerd op een literatuurstudie. Aangezien het onderwerp van theoretische aard is werd geen gebruik gemaakt van interviews of enquêtes. Een grondige literatuurstudie is voldoende om een goed onderzoek te verrichten en om tot zinvolle conclusies te kunnen komen.

Vervolgens wordt gebruik gemaakt van experimenten. De experimenten worden via een informatiesysteem uitgevoerd dat het memetisch algoritme en het gekozen vestigingsmodel implementeert. De experimenten worden verricht op specifiek geselecteerde datasets. Het doel van deze experimenten is na te gaan welke parameters een positief of negatief effect hebben op de kwaliteit van de resultaten.

1.5 Overzicht van de verdere opbouw

De literatuurstudie beslaat de drie volgende hoofdstukken. Het eerstvolgende hoofdstuk behandelt het *vestigingsprobleem*. De bedoeling is meer inzicht te verschaffen wat het vestigingsprobleem betreft. Verschillende aspecten die in het locatieprobleem aan bod kunnen komen worden besproken. Hierop volgend worden de vier klassieke discrete locatiemodellen bondig toegelicht. Het model dat geïmplementeerd wordt in het onderzoek, namelijk het *capacitated fixed charge model*, wordt wiskundig geformuleerd.

Het derde hoofdstuk bevat een overzicht van verschillende oplossingsmethoden. De focus ligt op oplossingsmethoden voor het gekozen model. Eerst worden exacte algoritmen besproken. Vervolgens komen heuristische oplossingsmethodes aan bod. Het laatste onderdeel behandelt de metaheuristieken.

Hoofdstuk vier leidt ons in in de wereld van de memetische algoritmes. Eerst worden de evolutionaire algoritmen verklaard. Vervolgens wordt de werking en de afkomst van memetische algoritmes uitgelegd. Tenslotte wordt het memetisch algoritme dat gebruikt wordt voor het onderzoek nader bepaald.

Het voorlaatste hoofdstuk, hoofdstuk vijf, begint met de uitleg over de werking van het gebruikte informatiesysteem. Hier wordt duidelijk gemaakt hoe alles is geïmplementeerd in het onderzoek. In het tweede deel wordt aangetoond met welke parameters zullen variëren. Het laatste deel geeft weer hoe de datasets gegenereerd worden.

Hoofdstuk zes tenslotte analyseert de resultaten. Eerst wordt de methode uitgelegd waarmee de resultaten geanalyseerd worden. Vervolgens wordt deze methode toegepast op de verkregen resultaten. Uiteindelijk worden hieruit conclusies getrokken.

2. Het vestigingsprobleem

Eerst wordt meer inzicht verschaft wat het vestigingsprobleem betreft. Ook de economische context wordt achterhaald. Vervolgens werpen we een blik op de vier klassieke locatiemodellen en wordt één model uitgewerkt: het *capacitated fixed charge facility location model*. Ook wordt kort aandacht besteed aan eventuele uitbreidingen van een *facility location model*.

2.1 Inleiding

Wat verstaat men onder een *facility*? Hansen (1987, p.1) geeft de volgende mogelijkheden: fabrieken, opslagplaatsen, scholen, hospitalen, administratieve gebouwen, filialen, militaire basissen, zieken- of brandweerwagendepots… Current et al. (2002, p. 83-84) geven tal van extra toepassingen van een *facility*, waaronder publieke zwembaden, bushaltes, fastfoodketens, luchthavens. Voor het gemak zullen wij in het vervolg het woord vestiging gebruiken. Deel 2.2 geeft meer informatie over deze verschillende mogelijkheden.

Waar moet de nieuwe vestiging worden geplaatst? Deze vraag behoort tot de strategische beslissingen van het management (Ballou, 1973, p. 550; Roodbergen, 2001, p. 5). Het is een belangrijke lange-termijn beslissing die vorm geeft aan het volledig *supply chain* systeem van een bedrijf. In deze wereld van economische globalisatie is het een belangrijk aspect geworden (Chase en Aquilano, p. 407).

De mate waarin een bedrijf bekwaam is om zijn goederen te produceren en effectief te verhandelen, of waarin een dienstenbedrijf bekwaam is om diensten van hoge kwaliteit te leveren, hangt gedeeltelijk af van de locatie van de (diensten)bedrijven in relatie tot andere vestigingen en tot de klanten. Terwijl het plaatsen van vestigingen een aanzienlijke uitgave vereist, creëert het in ruil kostbesparingen (o.a. transportkosten), een verbetering van de service of verbeterde competitieve voordelen (Coyle et al., 1992, p.426).

Huidige beslissingen betreffende *facility location* beïnvloeden de logistieke, marketing-, productie- en financiële kosten in de toekomst. Bovendien is het mogelijk dat een vestiging, goed geplaatst onder de huidige economische, competitieve en technologische omstandigheden, niet optimaal gevestigd is in de toekomstige omstandigheden (Coyle et al., 1992, p.425).

Coyle et al. (1992, p.428-430) geven een opsomming van belangrijke determinanten bij *facility location*. Het belang van de determinanten verschilt echter per industrie en per individueel bedrijf in de specifieke industrietak. Regionale determinanten kunnen onderscheiden worden van determinanten voor de specifieke ligging. In de eerste categorie kunnen volgende determinanten geplaatst worden: gunstig werkklimaat, nabijheid tot markten, levenskwaliteit, dichtstbijzijnde leveranciers, werkratio, reeds beschikbare vestiging of grond. De tweede categorie bevat determinanten zoals aanwezigheid van een spoorweg, een autosnelweg, speciale voorzieningen, landelijk gebied, aanwezigheid van water, transportmogelijkheden (lucht en truck). Op basis van deze determinanten kunnen potentiële vestigingen geselecteerd worden waarna een algoritme zal weergeven welke vestigingen dienen geopend te worden opdat een minimale kost verkregen wordt.

2.2 Modelkeuze

Voordat het model voor *facility location* van deze eindverhandeling wordt uitgewerkt, is het nuttig om de verschillende keuzes die gemaakt dienen te worden bij *facility location* kort toe te lichten. Vervolgens worden de vier klassieke modellen van de discrete klasse waartoe ons vestigingsprobleem hoort besproken. Uit deze vier modellen wordt ten slotte één model gekozen voor het onderzoek.

2.2.1 Modelklassen

In de literatuur vinden we volgende klassen terug (Hamacher & Nickel, 1998; Eiselt & Laporte, 1995, p. 152-156):

Continu of discreet

Twee belangrijke klassen zijn de continue en de discrete *facility location*. Continu wil zeggen dat eender welk punt in het gebied mag gekozen worden. Wanneer uit een vast aantal locaties van potentiële vestigingen moet gekozen worden, spreken we over *discrete facility location*. In deze eindverhandeling spitsen we ons toe op het **discrete** model, wat ook meer realistisch is. Voor een algemeen overzicht van continue locatiemodellen verwijzen we naar Plastria (1995).

Enkelvoudig of meervoudig

Voorts hebben we het onderscheid enkelvoudige-meervoudige *facility location*. De benaming zegt het zelf: bij enkelvoudige *facility location* wordt slechts één vestiging geplaatst, bij meervoudige worden meerdere vestigingen geplaatst. Wij zullen hier **meervoudige** *facility location* behandelen, mits een bedrijf zelden enkel één vestiging bezit.

Plaatsingsaspect of klantenaspect

Bij meervoudige *facility location* dient niet enkel het plaatsingsaspect van het probleem beschouwd te worden, maar moeten de klanten ook aan de vestigingen toegewezen worden. Wij beperken ons hier tot het **plaatsingsaspect** omdat dit voldoende is om de onderzoeksvraag de beantwoorden. Het klantenaspect wordt impliciet in het model opgenomen om te kiezen welke vestigingen geopend worden, maar de oplossingen zullen niet bij de resultaten weergegeven worden.

Vast of variabel aantal vestigingen

Het aantal te plaatsen vestigingen kan vast of variabel zijn. Als het aantal te plaatsen vestigingen om een bepaalde reden vastligt, wordt dit in het model vaak aangeduid met de letter p als prefix van de modelnaam. Hier wordt een **variabel aantal** te plaatsen vestigingen behandeld. Het is immers niet logisch om met een vast aantal te plaatsen vestigingen te werken als het model *capacitated* is (zie *infra*). Hoeveel vestigingen geplaatst zullen worden hangt af zowel de vaste als de variabele kosten.

Vestigingen met of zonder capaciteiten

Een vijfde en substantieel verschil ligt in de opname van de capaciteiten van de vestigingen. Indien het model de capaciteiten van de vestigingen in rekening brengt, betekent dit dat elke potentiële vestiging een capaciteit heeft, dewelke de maximumvraag is waaraan het kan voldoen. Indien het model geen rekening houdt met de capaciteiten, spreken we van onbeperkte capaciteiten en een simpel of *uncapacitated facility location* (Cornuejols et al., 1990, p. 120).

In deze eindverhandeling wordt wel rekening gehouden **met** de **capaciteiten** van de vestigingen, wat ook realistischer is. Een ziekenhuis bijvoorbeeld heeft slechts een beperkt aantal bedden. Voor *uncapacitated facility* location wordt verwezen naar de literatuur (Cornuejols et al., 1990).

Bediening vanuit één of meerdere vestigingen

Klanten kunnen ofwel door één vestiging ofwel door **meerdere vestigingen bediend** worden. In deze eindverhandeling wordt deze laatste mogelijkheid behandeld.

2.2.2 Discrete modellen

Zoals eerder vermeld, is de keuze uit continue en discrete modellen een basisonderscheid. De literatuur omvat dan ook een aantal wiskundige modellen voor *discrete facility location*. Daskin (1995, p. 92 – 302) bespreekt vier klassieke discrete *facility location* modellen: het *covering*, het centrum-, het mediaan-, en het *fixed charge location* model. Deze komen ook aan bod bij Owen en Daskin (1998) en Current et al. (2002, p. 81-118). Elk model zal nader worden bekeken en tenslotte zal één model gekozen worden voor het onderzoek.

Het **mediaanmodel** berekent de gemiddelde afstand die de klanten moeten afleggen om de vestiging te bereiken. Het mediaanmodel wordt voornamelijk gebruikt door private ondernemingen. Voor hen is het immers belangrijk dat ze goed bereikbaar zijn voor het grootste deel van de klanten. Ook publieke instellingen kunnen van het mediaanmodel gebruik maken.

Het ***covering* model** tracht een minimum aantal vestigingen te plaatsen om aan een bepaald serviceniveau te voldoen. De vestigingen dienen zodanig geplaatst te worden dat de afstand tussen elke klant en de vestiging de voorafbepaalde *covering*-afstand niet overschrijdt. Voor hulpdiensten is dit een aangewezen methode. Het eerste *covering location* model was het *set covering location* model. Voor elke klant dient voldaan te worden aan de beperking van maximale afstand tot de meest dichtbijgelegen vestiging. Omdat vaak niet voldoende middelen aanwijzig zijn om alle benodigde vestigingen te bouwen, volgde enkele jaren later het *maximum covering location model*. Het doel hierbij is de vestigingen zo te plaatsen dat het aantal niet-*gecoverde* klanten geminimaliseerd wordt. Niet-*gecoverde* klanten zijn klanten waarbij de afstand tot de dichtstbijzijnde vestiging groter is dan de maximale aanvaardbare *covering*-afstand.

Het **centrum-model** zorgt dat de maximale afstand tussen elke klant en haar meest dichtbijgelegen vestiging geminimaliseerd wordt. Voornamelijk publieke instellingen, zoals bibliotheken en scholen, maken gebruik van dit model. Het is immers belangrijk voor

publieke instellingen dat ze goed bereikbaar zijn voor iedereen. In tegenstelling tot het *covering* model wordt gewoonweg de langste afstand geminimaliseerd in plaats van een extra beperking in te stellen die de afstanden niet groter laat worden dan de vooropgestelde *covering*-afstand. Het *covering-* en het centrumlocatiemodel zijn dus maximale afstandsmodellen. Ze leggen de nadruk op het bedienen van de verste potentiële klant.

Deze voorgaande modellen concentreren zich op reisafstand of reistijd ter vervanging voor operationele kosten eens een vestiging geplaatst is. Hoewel beperkte geldmiddelen het aantal geplaatste vestigingen zouden kunnen bepalen, wordt slechts in één van die modellen (*set covering*) de vestigingskosten expliciet beschouwd. Bij het *fixed charge location* model wordt een vast bedrag geassocieerd met elke potentiële vestiging. Wanneer een bepaalde vestiging geopend wordt, komt de vaste kost van die vestiging bij in de totale kost. Deze benadering is realistischer dan de modellen waar enkel de reisafstand of –tijd beschouwd wordt.

Het ***fixed charge location* model** minimaliseert de totale kosten. Deze totale kosten bestaan uit twee componenten. De eerste component is de vaste kost die gerelateerd is aan het al dan niet openen van de vestiging. De kost van het openen van een vestiging wordt hier in rekening gebracht. De tweede component is de variabele kost van een vestiging naar een klant. Het model bepaalt het optimaal aantal vestigingen en de optimale locaties van de vestigingen, alsook de toewijzing van de klanten aan een de vestiging. Indien de vestigingen over een bepaalde capaciteit beschikken, bestaat de mogelijkheid dat een klant niet aan de dichtstbijzijnde vestiging wordt toegewezen. Wanneer geen sprake zou zijn van een capaciteitsbeperking worden de klanten wél aan de dichtstbijzijnde vestiging toegewezen. Dit geldt ook voor de eerste drie modellen. De capaciteitsbeperking is echter een meer realistische benadering van het vestigingsprobleem. Daarom zal het *capacitated fixed charge* model behandeld worden in het onderzoek van deze eindverhandeling.

Tot slot dient opgemerkt te worden dat de vier voorgaande modellen veronderstellen dat de klant rechtstreeks bediend wordt vanuit de vestiging. In de meeste gevallen is dit niet zo. *Location-routing* modellen beslissen niet alleen waar de vestigingen geplaatst worden en hoe de klanten aan deze vestigingen toegewezen worden, maar ook hoe de voertuigen geleid

moeten worden om de klanten te bedienen. Hier wordt geen rekening gehouden met de reisweg van de voertuigen mits dit buiten het bestek van de eindverhandeling valt.

Evenmin wordt in de voorgaande modellen rekening gehouden met de tijd. Dynamische modellen, in tegenstelling tot **statische** modellen, ontwerpen een vestigingsbeslissingsplan voor een gegeven planningsperiode waar veranderingen in de markt en in kosten worden verwacht. Een ander minpunt van de statische modellen is dat ze veronderstellen dat de parameters van het probleem met zekerheid gekend zijn. In de meeste *facility location* modellen heerst echter een aanzienlijke onzekerheid, vandaar het bestaan van de stochastische locatiemodellen. Mits dynamische modellen extreem moeilijk oplosbaar zijn, houden we het hier op statische locatiemodellen.

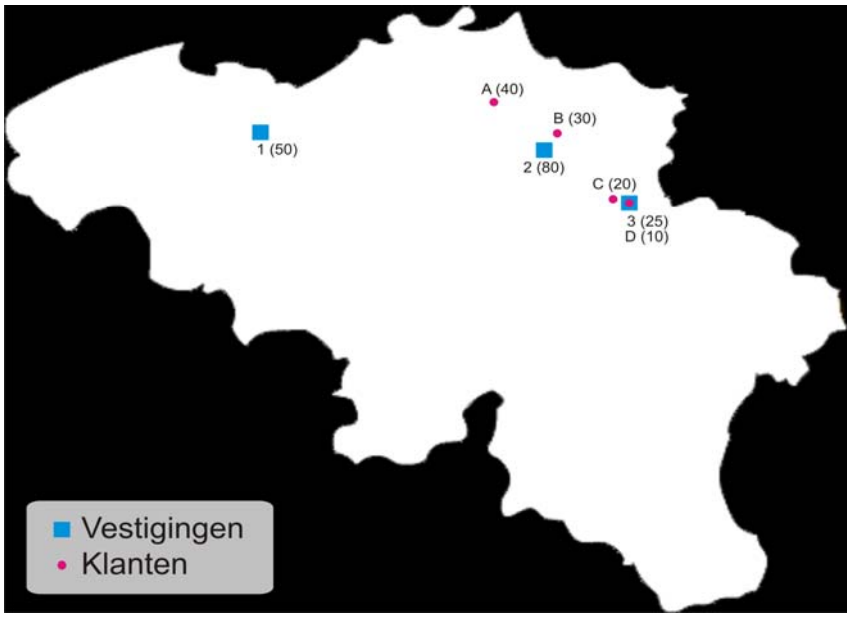
Voor de geïnteresseerde lezer wordt verder verwezen naar Daskin (1995, p. 92 – 302) en Current et al. (2002).

2.2.3 Het *capacitated fixed charge* model

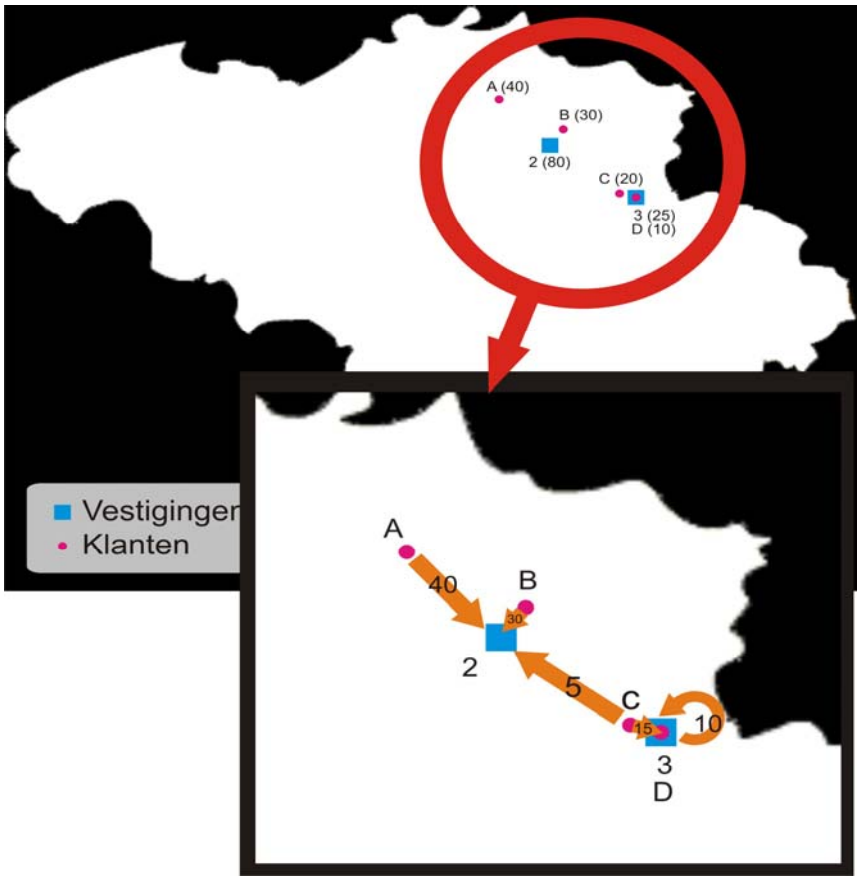
Voordat de wiskundige formulering gegeven wordt, wordt het model verduidelijkt aan de hand van een voorbeeld.

Voorbeeld van het capacitated fixed charge location model

Bedrijf XYZ heeft klanten in Herentals (A), Tessenderlo (B), Borgloon (C) en Tongeren (D) met een vraag van respectievelijk 40, 30, 20 en 10. Het bedrijf kan kiezen uit vestigingen te Gent (1), Diest (2) en Tongeren (3) met capaciteit van respectievelijk 50, 80 en 25. Veronderstel dat de variabele kost evenredig is met de af te leggen afstand tussen klant en vestiging en dat de vaste kost recht evenredig is met de capaciteit van de vestiging. De situatie wordt in figuur 2.1 voorgesteld.



Figuur 2.1: Voorbeeld van een vestigingsprobleem met 4 klanten en 3 potentiële vestigingen



Figuur 2.2: Voorbeeld van een vestigingsprobleem: de toewijzing van de klanten aan de geopende vestigingen

De klant te Tongeren (D) wordt hier volledig toegewezen aan de vestiging te Tongeren (3), terwijl de klant te Borgloon (C) slechts gedeeltelijk wordt toegewezen aan de vestiging te Tongeren (3) om de capaciteit van de vestiging niet te overschrijden. De overige vraag van 5 wordt aan de vestiging te Diest (2) toegewezen, samen met de volledige vraag van de klanten de Tessenderlo (B) en te Herentals (A). De vestiging te Gent (1) wordt niet geopend aangezien dan de variabele kosten te hoog zouden oplopen en de overige twee vestigingen over voldoende capaciteiten beschikken om aan de totale vraag te voldoen. De toewijzing wordt weergegeven in figuur 2.2.

Wiskundige formulering

In deze eindverhandeling wordt, zoals net vermeld, gewerkt met het *capacitated fixed charge location* model. Dit model bevat, in vergelijking met de andere modellen, een meer realistische weergave. Enerzijds wordt een vaste kost aan elke potentiële vestiging toegekend, anderzijds heeft elke potentiële vestiging een bepaalde capaciteit. De wiskundige formulering kan als volgt worden gegeven (Sridharan, 1995):

Stel dat men beschikt over *n* potentiële vestigingen en *m* klanten, dan kan het *capacitated fixed charge model*, anders gezegd het vestigingsprobleem met capaciteitsbeperking, geformuleerd worden als een *mixed integer programming* probleem (MIP-probleem). Het wiskundige model voor MIP is het lineaire programmeringsmodel waarbij bepaalde variabelen enkel gehele waarden mogen aannemen. De overige variabelen mogen ook niet-gehele waarden aannemen. Het adjectief lineair betekent dat alle wiskundige functies in dit model lineaire functies moeten zijn. Het woord *programming* verwijst niet naar computerprogrammering maar is eerder een synoniem voor planning. Met andere woorden, lineaire programmering houdt de planning van activiteiten in om een optimaal resultaat te bekomen (Hillier & Lieberman, 2005, p.25, 478). Het MIP-probleem voor het *capacitated fixed charge* model kan als volgt voorgesteld worden:

Minimaliseer $\sum_{j=1}^{n} f_j y_j + \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} x_{ij}$ (1)

Onderworpen aan:

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,\ldots,m \tag{2}$$

$$\sum_{i=1}^{m} d_i x_{ij} \le s_j y_j \qquad j = 1,\ldots,n \tag{3}$$

$$0 \le x_{ij} \le y_j \le 1 \qquad i = 1,\ldots,m;\ j = 1,\ldots,n \tag{4}$$

$$y_j = \{0,1\} \qquad j = 1,\ldots,n \tag{5}$$

$$\sum_{j=1}^{n} s_j y_j \ge \sum_{i=1}^{m} d_i \tag{6}$$

Waarbij gebruik gemaakt wordt van de volgende gegevens:

$I = \{1,\ldots,m\}$ = de verzameling van klanten

$J = \{1,\ldots,n\}$= de verzameling van de locaties van de potentiële vestigingen

f_j = vaste kost bij het plaatsen van een vestiging op de potentiële ligging j

s_j= de capaciteit van een vestiging op de potentiële ligging j

c_{ij} = totale transportkost om klant i te bedienen vanuit vestiging j

d_i = de vraag van klant i

en de volgende beslissingsvariabelen:

x_{ij}= het procentuele deel van de vraag van klant i dat door de vestiging op locatie j geleverd wordt

y_j= $\begin{cases} 1 & \text{indien een vestiging geplaatst wordt op locatie } j \\ 0 & \text{indien er geen vestiging geplaatst wordt op locatie } j \end{cases}$

De doelfunctie (1) minimaliseert de som van de totale transportkosten om de vraag te bedienen en de vaste *facility location* kosten. De eerste set termen in (1) wijst op de vaste kosten. De tweede set termen in (2) wordt vaak de gewogen-vraag-afstand genoemd en duidt op de variabele kosten. De eerste beperking (2) garandeert dat volledig aan de vraag van elke klant wordt voldaan. De volgende beperking (3) zorgt dat de capaciteiten van de geopende vestigingen niet overschreden worden en dat enkel geopende vestigingen klanten bedienen. Beperking (4) laat toe dat de vraag van een klant aan verschillende vestigingen kan worden toegewezen. Beperking (5) maakt y_j binair. Hierdoor laat beperking (4) ook toe dat zowel fracties van de vraag van een klant als de volledige vraag van een klant aan de geopende vestigingen kunnen worden toegewezen. Beperking (6) tenslotte stelt dat de totale capaciteit van de open vestigingen minstens zo groot moet zijn als de totale vraag van de klanten.

Het *facility location* probleem bestaat uit twee beslissingen die simultaan genomen worden. De belangrijkste beslissing bestaat erin te bepalen welke vestigingen geopend worden en welke vestigingen gesloten blijven. Eens de verzameling van open vestigingen vastligt, blijft een zuiver transportprobleem over. Aan de hand van de vraag van elke klant en de capaciteit van elke vestiging wordt in het transportprobleem de hoeveelheid te vervoeren goederen van elke vestiging naar elke klant bepaald, waarbij de transportkosten geminimaliseerd worden. De transportkost wordt verondersteld recht evenredig te zijn met de getransporteerde hoeveelheid. (Sörensen, 2003, p. 145)

Tot slot dient aangegeven te worden dat in deze eindverhandeling nooit op een bestaand aantal vestigingen wordt verder gebouwd. In het gebied waar de vestigingen dienen geplaatst te worden, liggen nog geen vestigingen van dezelfde onderneming.

Als hierna verwezen wordt naar het vestigingsprobleem of ons vestigingsprobleem, moet men dus dit model voor ogen hebben. Ook deze symbolen zullen verder gebruikt worden met dezelfde betekenis.

3. Oplossingsmethoden

Vooreerst wordt een duidelijk overzicht gegeven over oplossingsmethoden: algoritmes, heuristieken en metaheuristieken. Het tweede onderdeel, namelijk de exacte algoritmes, bespreekt de *branch & bound* procedure toegepast op ons vestigingsprobleem. Het derde onderdeel geeft heuristische oplossingsmethoden weer. Het vierde onderdeel behandelt de metaheuristieken. Voor zowel de heuristische oplossingsmethoden als de metaheuristieken wordt nader ingegaan op oplossingsmethoden die in de literatuur reeds zijn toegepast op ons probleem. Extra aandacht wordt besteed aan de metaheuristieken aangezien de memetische algoritmes hiertoe behoren.

3.1 Inleiding

Na het opstellen van het wiskundig model in deel 2.2 dient een procedure ontwikkeld te worden om oplossingen voor dit probleem af te leiden. Via een algoritme, dit is een systematische oplossingsprocedure, is het mogelijk een optimale oplossing te bekomen. Wel moet opgemerkt worden dat die optimale oplossing enkel optimaal is wat het model betreft. Het model is eerder geïdealiseerd dan een exacte voorstelling van het echte probleem. Met echte problemen zijn gewoonweg te veel onzekerheden en onvoorspelbare factoren geassocieerd. (Hillier & Lieberman, 2005, p. 15-16)

Soms zijn de problemen zo ingewikkeld dat het onmogelijk is om een optimale oplossing te vinden met een algoritme. In dat geval blijft het belangrijk om een goede toelaatbare oplossing te zoeken die op zijn minst de optimale oplossing benadert. Heuristische methodes bieden in dit geval goede oplossingen. Een heuristische methode is een procedure die voor een specifiek probleem tot een zeer goede toelaatbare oplossing kan leiden, maar niet noodzakelijk tot een optimale oplossing. De procedure is normalerwijze voldoende efficiënt om zeer grote problemen snel aan te pakken. Elke procedure is gewoonlijk ontworpen om eerder een specifiek probleemtype op te lossen dan een reeks uiteenlopende toepassingen. (Hillier & Lieberman, 2005, p. 617)

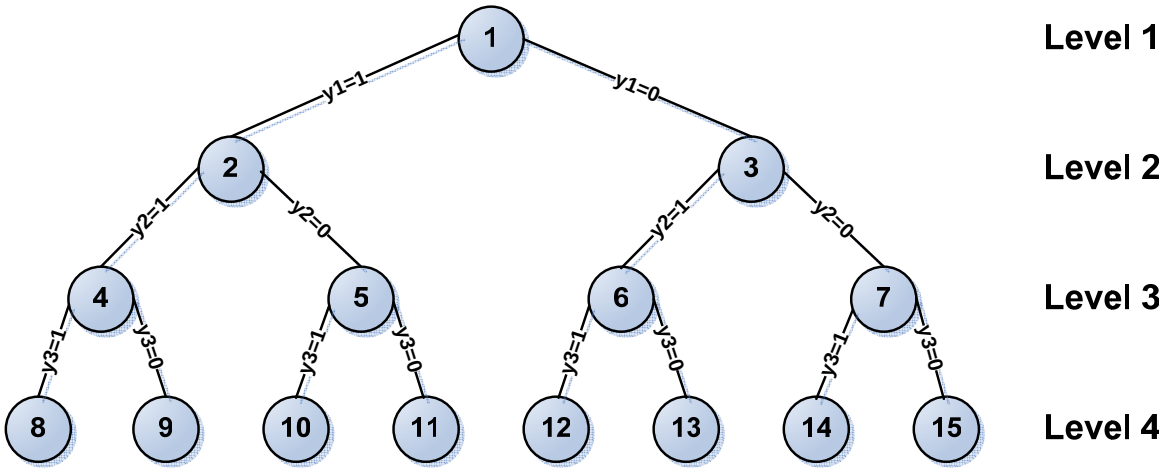
Jarenlang betekende dit dat een operationele onderzoeker van nul zou moeten beginnen om een heuristische methode te ontwikkelen voor het betreffende probleem wanneer een algoritme voor het probleem niet beschikbaar was. Dit is allemaal veranderd in de recentere jaren dankzij de ontwikkeling van krachtige metaheuristieken. Een metaheuristiek is een algemene oplossingsmethode die zowel een algemene structuur als strategische richtlijnen voorziet om een specifieke heuristische methode te ontwikkelen om een bepaald soort probleem te passen. Metaheuristieken brengen de snelheid van eenvoudige heuristieken en de nauwkeurigheid van optimale algoritmes in evenwicht. (Hillier & Lieberman, 2005, p. 617 en Sörensen, 2003, p. 28)

3.2 Exacte algoritmes

3.2.1 Branch and bound & enumeration tree

Van een klein aantal technieken is slechts bewezen dat ze effectief discrete locatieproblemen optimaal kunnen oplossen. De meest fundamentele techniek is die van *enumeration* (opsomming). *Enumeration* is een systematische benadering die alle 2^n mogelijke locatiebeslissingen opsomt. Deze opsomming van alle 2^n locatiebeslissingen kan tot stand gebracht worden via een opsommingsboom (figuur 3.1). (Francis et al., 1983)

De knooppunten op elk niveau van de boom komen met een beslissing over een bepaalde potentiële locatie overeen, behalve de knooppunten op niveau 4 (n + 1). Deze onderste rij knooppunten stemt overeen met volledige beslissingen: welke vestigingen dienen uiteindelijk geopend te worden en welke niet. De overige niveaus, namelijk 1, 2 en 3 (= 1,…,n), bevatten knooppunten die overeenstemmen met gedeeltelijke beslissingen. Het is relatief gemakkelijk om een computerprogramma te schrijven dat zulke opsommingen uitvoert. (Francis et al., 1983)



Figuur 3.1: Voorbeeld van een opsommingsboom met $n = 3$

Algoritmes die gebaseerd zijn op opsomming dienen niet noodzakelijkerwijze alle 2^n oplossingen te evalueren. Sommige knooppunten kunnen expliciet vermeden worden indien het mogelijk is vast te stellen dat de optimale oplossing van een kandidaat-probleem[1] niet de optimale oplossing van het oorspronkelijke probleem zal zijn. Het kandidaat-probleem kan bijvoorbeeld geen toelaatbare oplossing hebben. (Sridharan, 1995)

Bij knooppunt 7 van het voorbeeld van de opsommingsboom zou het kandidaat-probleem bij wijze van voorbeeld niet over voldoende capaciteit beschikken om aan de totale vraag te voldoen. Knooppunten 14 of 15 moeten bijgevolg niet verder beschouwd worden. Het kandidaat-probleem (het knooppunt) is *fathomed*[2]. Dit wil zeggen dat het niet verder moet beschouwd worden. In dit geval is *fathoming* gebaseerd op een toelaatbaarheidstest. (Francis et al., 1983)

Buiten de toelaatbaarheidstest kan een alternatieve test onderscheiden worden waardoor een kandidaat-probleem *fathomed* kan zijn. Die test probeert te bewijzen dat de best mogelijke oplossing van het kandidaat-probleem niet beter is dan de beste oplossing die tot nu gevonden is. Deze laatste beste oplossing wordt de *incumbent solution* genoemd. Aangezien bij ons

[1] De beslissingen die reeds gemaakt zijn bij een knooppunt verschillend van het knooppunt op niveau 1, definiëren een kandidaat-probleem. Een kandidaat-probleem is een locatieprobleem afkomstig van het oorspronkelijke locatieprobleem waarbij enkele beslissingen reeds gespecificeerd zijn.
[2] to fathom = doorgronden

vestigingsprobleem getracht wordt de totale kost te minimaliseren, voorziet de *incumbent solution* een bovengrens voor de best mogelijke oplossingswaarde. Indien kan aangetoond worden dat de beste oplossingswaarde van het kandidaat-probleem de bovengrens overschrijdt, dient dat kandidaat-probleem niet verder beschouwd te worden. Deze *fathoming* test wordt *bounding* genoemd. (Hillier & Lieberman, 2005, p. 505-509)

Het probleem met *bounding* bestaat erin dat de kandidaat-problemen zelf moeilijke discrete optimalisatieproblemen zijn en bijgevolg niet direct oplosbaar zijn. Indien dit immers mogelijk was, zou *enumeration* op de eerste plaats overbodig zijn. *Bounding* gebruikt dan een 'superoptimale' oplossingswaarde in plaats van een optimale oplossing van het kandidaat-probleem. Die 'superoptimale' oplossingswaarde wordt verkregen door een relaxatie van het kandidaat-probleem op te lossen.

Een geldig relaxatieprobleem, RP, bevat de volgende eigenschappen:

- een toelaatbare oplossing van KP[3] is ook toelaatbaar voor RP,
- de optimale oplossingswaarde voor RP, opt(RP), is tenminste zo klein als de optimale oplossingswaarde voor KP, opt(KP).

Zodoende, als opt(RP) groter is dan de bovengrens, dan is opt(KP) duidelijk groter dan de bovengrens. Het is onnodig om KP verder te beschouwen. Opt(RP) wordt vaak de *benedengrens* genoemd, aangezien opt(KP) groter of gelijk is dan/aan opt(RP). (Francis et al., 1983)

De eenvoudigste relaxatie is de lineaire programmeringsrelaxatie die verkregen wordt door de beperking (5) van het vestigingsprobleem, namelijk $y_j = \{0,1\}$; ($j = 1,\ldots,n$), te vervangen door de continue tegenhanger, $0 \leq y_j \leq 1$. Deze relaxatie werkt goed met sommige formuleringen, maar wordt beperkt door de grootte van het lineair programmeringsprobleem dat opgelost dient te worden. (Sridharan, 1995 en Francis et al., 1983)

[3] kandidaat-probleem

De combinatie van *enumeration* met *fathoming* tests wordt *branch and bound* genoemd. Voor een verdere uitwerking van het *branch and bound* algoritme voor ons vestigingsprobleem kan verwezen worden naar Sridharan (1995), Francis et al. (1983) en Akinc en Khumawala (1977).

De *branch and bound* methode werkt voor ten minste enkele voorbeelden van de meeste locatiemodellen en is typisch enkel bruikbaar voor kleine problemen. Locatiemodellen van realistische grootte kunnen gemakkelijk duizenden of honderdduizenden beperkingen en variabelen hebben. Het gebruik van het *branch and bound* algoritme leidt dan vaak tot onaanvaardbaar lange rekentijden en neemt een groot beslag op het computergeheugen, terwijl succes niet gegarandeerd is. De reden hiervoor is dat zelfs de meest basismodellen geclassificeerd zijn als *NP-hard*[4]. (Current et al., 2002, p. 101)

Hierdoor moeten andere methodes bedacht worden om optimale oplossingen te identificeren of, indien men hier niet in slaagt, zeer goede oplossingen te vinden. In het volgende deel zullen daarom heuristische oplossingsmethoden besproken worden.

3.3 Heuristische oplossingsmethoden

Soms is het mogelijk dat exacte algoritmes een probleem niet kunnen oplossen doordat het probleem te groot is en het moet rekening houden met veel beperkingen. Waar men exacte algoritmes kan gebruiken om middelgrote problemen (50 vestigingen – 50 klanten) op te lossen, met een redelijke computerinspanning, zijn heuristieken nodig om problemen met enkele honderden vestigingen en klanten op te lossen. Twee *greedy heuristics*, namelijk ADD en DROP, worden hier besproken. De vestigingen worden één per één geopend (ADD) of gesloten (DROP). Eens een beslissing vastligt over een welbepaalde potentiële vestiging, kan die niet gewijzigd worden. Voorts kan een tweede categorie, namelijk *interchange heuristics*, vermeld worden. Deze heuristieken trachten de beslissingen gemaakt bij ADD of DROP te verbeteren door een ‘opschudding’. Een typische opschuddingsmethode is de *bump and shift* routine van Kuehn & Hamburger (1963) voor een SPLP (*single plant location problem*). De

[4] *NP hard*-problemen zijn wiskundig zeer moeilijk oplosbare problemen

alternate location allocation (ALA) en de *vertex substitution method* (VSM) zijn voorbeelden van *interchange heuristics* (Sridharan, 1995). Een derde type heuristieken staan bekend onder *Langragian heuristics*, hierbij wordt een *Langrangian* relaxatie voor het vestigingsprobleem opgelost. Hiervoor wordt verwezen naar de literatuur (Cornuejols et al., 1991).

Greedy heuristics

Zoals reeds vermeld behandelen we twee verschillende *greedy heuristics*, namelijk de ADD-procedure en de DROP-procedure. De benaming zelf geeft veel informatie over de procedure. Bij beide procedures wordt gewerkt met drie subverzamelingen van J, namelijk J0, J1 en JØ. In subverzameling J0 zitten de potentiële vestigingen die niet geopend zullen worden. Elke y_j in deze subverzameling heeft de waarde 0. De potentiële vestigingen y_j die zich in J1 bevinden, worden geopend en hierbij gelijkgesteld aan 1. Voor de vestigingen die in subverzameling JØ zitten is nog geen beslissing genomen, met andere woorden alle y_j's $\in$ JØ zijn nog onbepaald. (Jacobsen, 1983 en Sridharan, 1995)

De ADD-procedure gaat uit van een situatie dat alle potentiële vestiging en gesloten zijn. Subverzamelingen J1 en JØ zijn hier leeg. De ADD-procedure begint echter met een ontoelaatbare oplossing. Dit wordt opgelost door een supervestiging te creëren die een capaciteit gelijk aan de totale vraag heeft en die hoge transportkosten naar de klanten hanteert. Via een zekere formule wordt stapsgewijs bepaald welke vestiging het best geopend wordt. De formule berekent de grootste besparingen die gerealiseerd kunnen worden. Zo wordt de vestiging gekozen die de grootste besparingen met zich meebrengt. Van zodra de supervestiging kan gesloten worden, dienen geen extra vestigingen geopend te worden. (Sridharan, 1995)

De DROP-procedure heeft een tegengestelde werking aan die van de ADD-procedure. Alle vestigingen zijn geopend bij aanvang van de DROP-procedure, J0 en JØ zijn leeg. Stapsgewijs worden die vestigingen gesloten waardoor de grootste besparingen worden verkregen. (Jacobsen, 1983)

Beide procedures brengen één groot nadeel met zich mee. Eens een vestiging gesloten (DROP) of geopend (ADD) is, blijft deze vestiging gesloten of geopend. Heuristieken die hierop verbeteringen realiseren worden *interchange heuristics* genoemd. (Sridharan, 1995)

Interchange heuristics

Hoewel de *greedy heuristics* effectief een toelaatbare oplossing met matige rekeninspanning kan bereiken, kan niet gegarandeerd worden dat goede oplossingen verschijnen. *Improvement heuristics* beginnen met een toelaatbare oplossing en trachten deze te verbeteren. *Interchange heuristics* zijn hier een voorbeeld van. Hier wordt de ALA-procedure nader toegelicht. Het basisschema van de ALA-procedure wordt als volgt weergegeven (Sridharan, 1995):

(i) Verkrijg een (goede) toelaatbare oplossing, bijvoorbeeld met de ADD- of DROP-procedure
(ii) Veroorzaak enige opschudding in de oplossing
(iii) Optimaliseer de oplossing opnieuw
(iv) Herhaal (ii) en (iii) tot de oplossing niet wijzigt

De opschudding bestaat gewoonlijk uit een geforceerde ADD (of DROP) iteratie. De nieuwe optimalisatie in stap (iii) betreft geen echte optimalisatie maar eerder een DROP (of ADD) iteratie. Het houdt het tegenovergestelde van de opschuddingsingreep in. De implementatie van ALA op de computer roept enkele vragen op aangaande het exact ontwerp van de verschillende stappen. Voor de VSM verwijzen we naar de literatuur (Jacobsen, 1983).

3.4 Metaheuristieken

Jaramillo et al. (2002) wijzen erop dat het *fixed charge location* model een NP-hard, dit wil zeggen moeilijk berekenbaar met de computer, combinatorisch optimalisatieprobleem[5] is. In de praktijk moeten beslissingnemers die zo een probleem onder ogen zien zich vaak tevreden

[5] Het domein van combinatorische optimalisatie betreft de optimalisatieproblemen waar de verzameling toelaatbare oplossingen discreet is of kan beperkt worden tot een discrete verzameling. Het doel is om de best mogelijke oplossing te vinden.

stellen met een oplossing die dichtbij de optimale oplossing ligt en die relatief snel gevonden wordt. Gewone heuristieken leveren echter vaak geen voldoende kwalitatieve oplossing. Vandaar het gebruik van metaheuristieken, waar de rekensnelheid van gewone heuristieken en de nauwkeurigheid van optimale algoritmen gebalanceerd worden (Sörensen, 2003, p. 28). Volgens Sörensen (2003, p. 27) is metaheuristieken een overkoepelende naam voor een verzameling optimalisatietechnieken die ruwweg in drie klassen kan verdeeld worden: **omgevingsgebaseerde**, **populatiegebaseerde** en **hybride** metaheuristieken.

Omgevingsgebaseerde metaheuristieken gaan verder dan de gewoonlijke *local search*[6] algoritmen. Hun doel is om aan deze lokale optima te ontsnappen. Bekende omgevingsgebaseerde metaheuristieken zijn *tabu search* en *simulated annealing*. **Populatiegebaseerde** metaheuristieken werken met een populatie oplossingen, die kan verbeterd worden door combinaties. Genetische algoritmen zijn populaire populatiegebaseerde metaheuristieken. De **hybride** metaheuristieken tenslotte combineren aspecten van de voornoemde categorieën. Dit leidt tot heuristieken van hoge kwaliteit. Een voorbeeld is een genetisch algoritme dat *tabu search* gebruikt om de individuele oplossingen in de populatie te verbeteren. (Sörensen, 2003, p.29-30)

In dit deel worden drie technieken en een eventuele toepassing van die technieken op ons probleem besproken. Vervolgens zullen de drie technieken worden vergeleken.

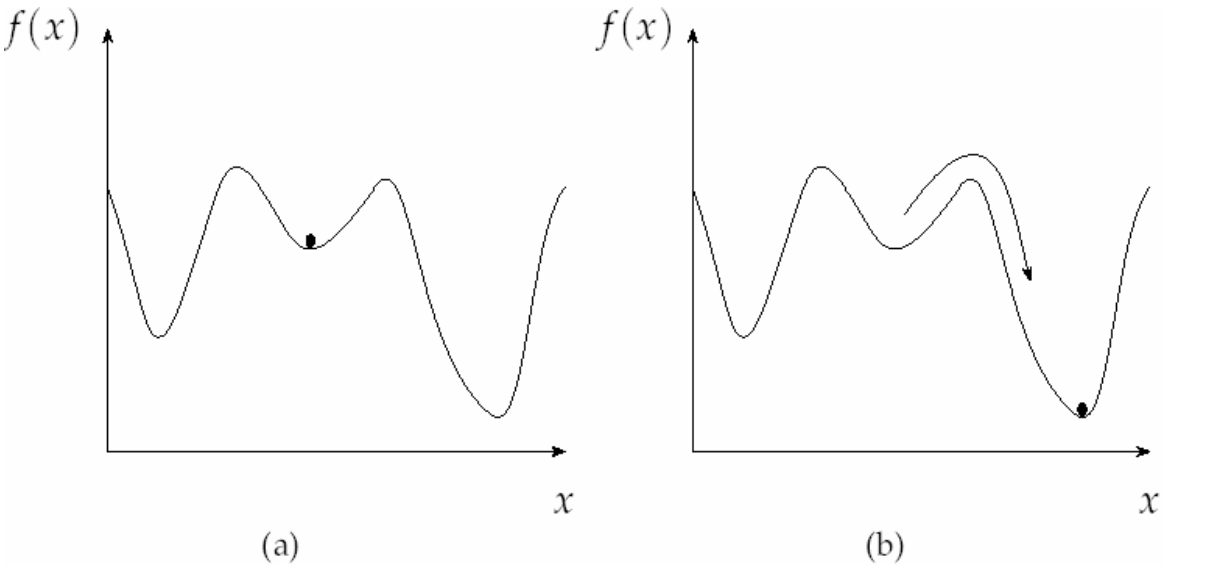
OMGEVINGSGEBASEERDE METAHEURISTIEKEN

3.4.1 Tabu Search (TS)

Tabu search is een veelgebruikte metaheuristiek. De techniek mijdt lokale optima, zoals ook *simulated annealing* (zie 3.4.2). Dit doet het door bij de *local search* procedure niet te eisen dat elke nieuwe proefoplossing beter is dan de vorige proefoplossing. *Tabu search* kan de zoektocht verder zetten door niet-verbeterende zetten toe te laten tot de beste oplossingen in de buurt van het lokale optimum. Indien een punt bereikt wordt waar betere oplossingen

[6] *Local search* is een beperkte zoektocht in de verzameling toelaatbare oplossingen van een optimalisatieprobleem. Bij lokaal search heerst het gevaar om in lokale optima terecht te komen.

gevonden kunnen worden in de buurt van de huidige proefoplossing, wordt de lokale verbeteringsprocedure opnieuw toegepast om een nieuw lokaal optimum te zoeken. (Hillier & Lieberman, 2005, p. 625)



Figuur 3.2: Ontsnappen van een lokaal optimum (minimum) via een metaheuristiek (Sörensen, 2003, p. 29)

Indien van dit lokale optimum weg geëvolueerd wordt, bestaat het gevaar dat het proces terug zal keren naar hetzelfde lokale optimum. Om dit te vermijden verbiedt *tabu search* tijdelijk zetten die zouden terugkeren naar een oplossing die we recent gehad hebben. Een tabulijst legt verboden zetten vast zodat we niet in een oneindige lus terechtkomen. Dit gebruik van een geheugen om de zoektocht te begeleiden via het gebruik van tabulijsten om de recente geschiedenis van de zoektocht te onthouden is een onderscheidend kenmerk van *tabu search*. (Hillier & Lieberman, 2005, p. 625)

Tabu search probeert aan de lokale optima te ontsnappen door gebruik te maken van geheugenstructuren. Zowel korte termijn als lange termijn geheugen wordt gebruikt en beide types hebben elk hun eigen speciale strategieën (Sörensen, 2003, p. 31). Nu we het *tabu search*-algoritme beter kennen, kunnen we kijken naar de toepassing op ons vestigingsprobleem.

Dankzij de speciale structuur van het *facility location* probleem kunnen de oplossingen gemakkelijk gecodeerd worden. Deze speciale structuur is te wijten aan het feit dat twee gerelateerde beslissingen genomen worden: de plaatsing van de vestigingen en de toewijzing van de vraag aan de open vestigingen. (zie *supra*) Deze speciale structuur laat toe om het *facility location* probleem in twee stadia te splitsen. Het eerste stadium legt de binaire variabelen y_j vast. Het tweede stadium bepaalt de hoeveelheid van het product geleverd aan klant *i* vanuit vestiging *j*.

Sörensen (2003, p.147-150) heeft bij zijn onderzoek een 'zet' gedefinieerd die enkel toelaat om één potentiële vestiging te openen of sluiten tijdens het *tabu search* algoritme. Deze zet wordt bepaald door een eenvoudige *local search* heuristiek: *steepest descent* of *first improving*. De *steepest descent* methode sluit vanaf de huidige oplossing elke open vestiging en opent elke gesloten vestiging om zo uiteindelijk de zet die de grootste stijging in de doelfunctie teweegbrengt, uit te voeren. De *first improving move* heuristiek voert de eerste *move* uit die de huidige oplossing verbetert. De *moves* worden onderzocht in de volgorde waarin de vestigingen verschijnen in de probleemdefiniëring.

Om verder te gaan dan lokale optima worden twee types geheugen gebruikt: korte termijn geheugen (de tabulijst) en lange termijn geheugen (de frequentielijst). De tabulijst wordt gebruikt om te voorkomen dat we in hetzelfde gebied van een lokaal optimum blijven zoeken.. Een vast aantal van de laatste oplossingen wordt daarom steeds in de tabulijst opgeslagen. Als een bepaalde oplossing in de tabulijst zit, is het verboden om een *move* uit te voeren waardoor de huidige oplossing een oplossing van de tabulijst zou zijn. De frequentielijst houdt voor elke potentiële vestiging bij gedurende welk aantal stappen het geopend was. Deze lijst wordt gebruikt om de zoektocht op een nieuw spoor te zetten door een potentiële vestiging te openen die tot op dat moment het minst aantal keren geopend was.

Resultaten van het onderzoek wijzen erop dat deze metaheuristiek uitstekend is voor eerder kleine voorbeelden in de literatuur. Onderzoek bij grotere voorbeelden dient nog uitgevoerd te worden. Ook werd aangetoond dat de *steepest descent move* methode aanzienlijk beter resultaten leverde dan het *first improving move* mechanisme.

3.4.2 Simulated annealing (SA)

Kort samengevat is *tabu search* de huidige helling afdalen in de steilste richting tot het dal is bereikt en daarna beginnen zo min mogelijk te stijgen terwijl we een andere helling zoekt om af te dalen. Het nadeel is dat veel tijd (stappen) wordt besteed aan het afdalen van elke helling terwijl minder tijd gespendeerd wordt om de grootste helling te zoeken. (Hillier & Lieberman, 2005, p. 635)

De benadering in *simulated annealing* focust hoofdzakelijk op het zoeken naar de grootste helling. Aangezien de grootste helling zich eender waar in het toelaatbare gebied kan bevinden, ligt de vroegtijdige nadruk op stappen nemen in willekeurige richtingen om zoveel mogelijk van het toelaatbare gebied te onderzoeken. Omdat de meeste aanvaarde stappen neerwaarts zijn, zal de zoektocht geleidelijk neigen naar deze delen van het toelaatbare gebied die de grootste hellingen bevatten. Hiervoor legt het zoekproces geleidelijk meer de nadruk op neerwaarts afdalen door een groeiende proportie stappen te verwerpen die opwaarts gaan (Hillier & Lieberman, 2005, p. 635). *Simulated annealing* zoekt niet naar de beste oplossing in de buurt van de huidige proefoplossing, maar kiest willekeurig een oplossing in die buurt. Als deze oplossing een betere waarde heeft dan de huidige proefoplossing, dan wordt deze de volgende proefoplossing. Indien dit niet het geval is, wordt de nieuwe oplossing slechts de nieuwe proefoplossing met een bepaalde kans. Deze kans vermindert met de tijd en met de grootte van de afwijking van de waarde van de nieuwe oplossing tegenover de huidige oplossing. Naarmate de procedure vordert, worden daardoor enkel verbeterende oplossingen aanvaard (Pirlot, 1996). Als we voldoende tijd hebben, zal het proces vaak het dal van de grootste helling bereiken en afdalen (Hillier & Lieberman, 2005, p. 635).

Kort samengevat beweegt elke iteratie van het '*simulated annealing*'-zoekproces van de huidige oplossing naar een onmiddellijke buur in de lokale omgeving van deze oplossing, net als in *tabu search*. *Simulated annealing* verschilt met *tabu search* door de manier waarop een onmiddellijke buur wordt geselecteerd om de volgende huidige oplossing te zijn. (Hillier & Lieberman, 2005, p. 636)

Simulated annealing vindt de oorsprong in thermodynamica en metallurgie. Wanneer gesmolten ijzer langzaam genoeg wordt gekoeld neigt het hard te worden in een structuur van minimale energie. Dit ontgloeiingsproces (*annealing process*) wordt nagebootst bij de *local search*-strategie. In het begin wordt bijna elke zet aanvaard. Dit laat ons toe om de oplossingsruimte te verkennen. Dan wordt de 'temperatuur' geleidelijk aan verlaagd wat betekent dat het algoritme meer en meer selectief wordt in het aanvaarden van nieuwe oplossingen. Op het einde worden enkel zetten aanvaard die de waarde van de huidige oplossing echt verbeteren. (Pirlot, 1996)

POPULATIEGEBASEERDE METAHEURISTIEKEN

3.4.3 Genetische algoritmes (GA)

Genetische algoritmes zijn mogelijk de meest bekende metaheuristiek. Ze steunen op het concept van de biologische evolutietheorie van Charles Darwin. In tegenstelling tot *simulated annealing* en *tabu search* start een genetisch algoritme met een populatie oplossingen. Door eenvoudige genetische operatoren wordt geprobeerd aan de hand van oude oplossingen nieuwe oplossingen te genereren zodanig dat de populatie geleidelijk verbetert. (Reeves, 1996, p. 17)

Zoals eerder vermeld, behoren de genetische algoritmes tot de klasse van de populatiegebaseerde metaheuristieken. Vooraleer we de werking uitleggen zullen we de herkomst van de genetische algoritmes uitdiepen. Zo is de werking gemakkelijker verstaanbaar.

Elke planten- en dierensoort heeft een grote individuele variatie. Darwin observeerde dat de individuen met variaties die een overlevingsvoordeel met zich meebrengen door verbeterde aanpassing aan de omgeving, het meest waarschijnlijk zullen overleven in de volgende generatie. Dit fenomeen staat beter bekend onder '*survival of the fittest*'.

Het evolutieproces en de natuurlijke selectie worden in de werking geïmplementeerd. Bij eender welke soort die zich reproduceert doormiddel van seksuele reproductie erven de kinderen chromosomen van elk van de twee ouders. De genen in de chromosomen bepalen de individuele kenmerken van het kind. Een kind dat betere kenmerken van de ouders erft heeft een grotere overlevingskans en kan op zijn/haar beurt ouder worden die enkele van zijn/haar kenmerken naar de volgende generatie overbrengt. De populatie zal zo lichtjes verbeteren in de tijd. Ook mutaties spelen een rol in het evolutieproces. Een mutatie komt af en toe voor en verandert de kenmerken van een chromosoom dat een kind erft van een ouder. Hoewel meeste mutaties geen effect hebben of nadelig zijn, vertonen sommige mutaties gewenste verbeteringen. (Hillier & Lieberman, 2005, p. 644-645)

Na het coderen van de oplossingen in strings en de variabelen te hebben vastgelegd (populatiegrootte, selectiemechanisme,…) volgt het genetisch algoritme de volgende stappen (Hillier & Lieberman, 2005, p. 645):

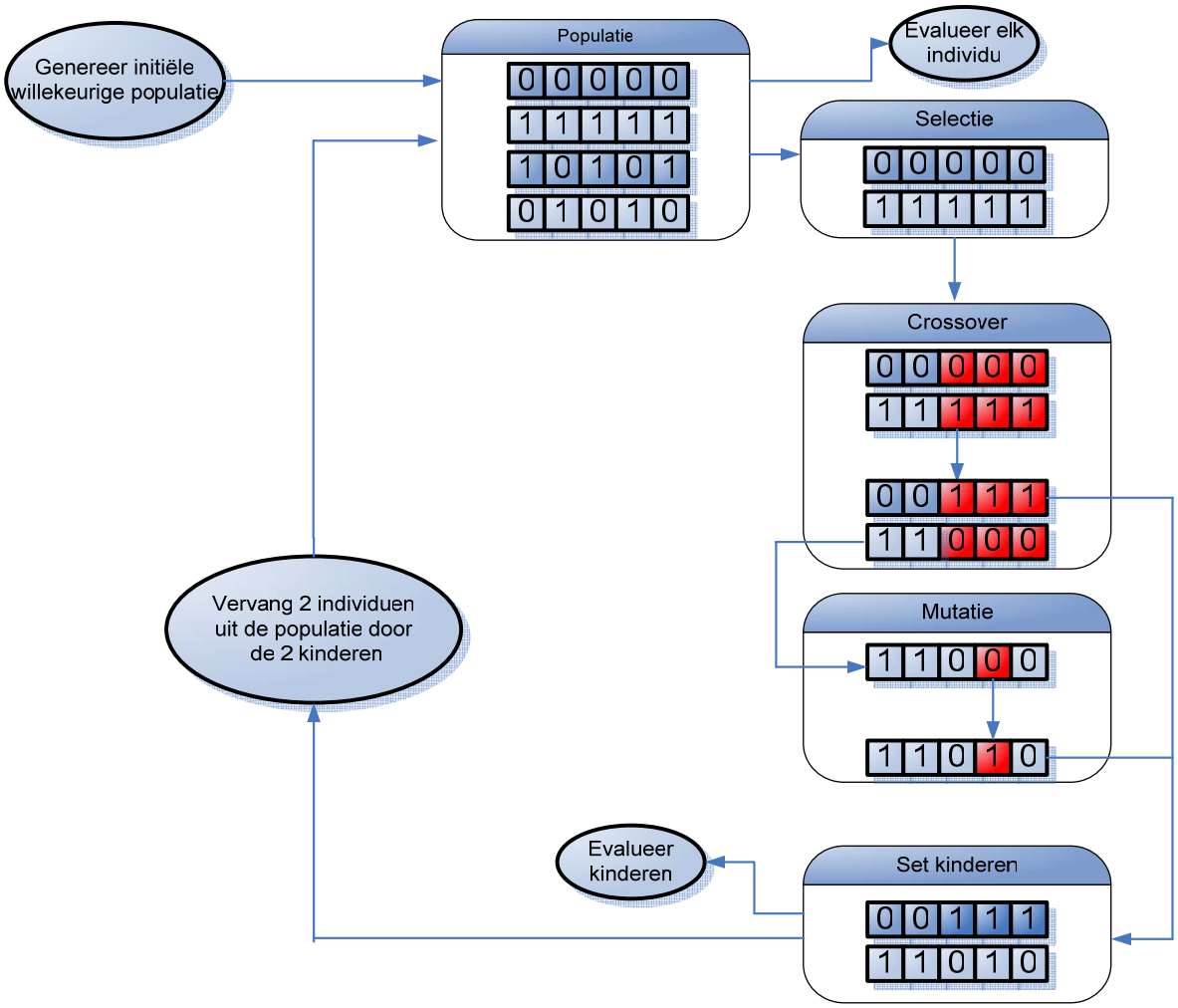
Initialisatie Genereer *at random* een initiële populatie en evalueer elke oplossing aan de hand van de doelfunctie.

Stap 1 Selecteer een even aantal oplossingen aan de hand van het selectiemechanisme. Deze oplossingen zijn de zogenaamde ouders.
Stap 2 Koppel de ouders *at random* en laat ze twee kinderen tot leven brengen via *crossover*..
Stap 3 Pas eventueel een mutatie toe.
Stap 4 Voeg de kinderen bij de populatie en verwijder een gelijk aantal oplossingen uit die populatie via een selectiemechanisme.

Stopregel Gebruik een bepaalde stopregel, zoals een vast aantal stappen, een vaste hoeveelheid CPU[7]-tijd, of een vast aantal opeenvolgende stappen zonder enige verbeteringen in de beste oplossing tot dan gevonden. De finale oplossing is de beste oplossing gevonden in eender welke stap.

[7] *Central processing unit*, de processor van een computer

In figuur 3.3 wordt het genetisch algoritme verduidelijkt aan de hand van een voorbeeld.



Figuur 3.3: De algemene structuur van genetische algoritmen (Jaramillo et al., 2002)

Jaramillo et al. (2002) gebruiken genetische algoritmen om verschillende locatieproblemen op te lossen. Een klein deel van het onderzoek werd gewijd aan ons vestigingsprobleem. Na een aantal parameters vast te leggen (grootte van populatie, procedure voor de selectie van de ouders,…), implementeerden Jaramillo et al. (2002) het volgend genetisch algoritme voor hun onderzoek:

Tabel 3.1: Een genetisch algoritme voor ons vestigingsprobleem (Jaramillo, 2002)

Step 1	Set *t*=0.
Step 2	Generate initial population, *P(t)* randomly.
Step 3	Evaluate each of the strings in *P(t)* according to the kind of problem being solved.
Step 4	While (number of generations ≤ maximum value) or (improvement objective function value ≤ 10-5) do.
Step 5	Set *t*=*t*+1.
Step 6	Select two solutions *P1* and *P2* from the population using binary tournament selection.
Step 7	Apply genetic operators to strings *P1* and *P2.*
	If Crossover: Combine *P1* and *P2* to form a offspring *O1* using the fusion crossover operator. If *O1* is identical to any of its parents, then apply mutation operator to the parent with the best fitness.
	If Mutation: Apply mutation operator to the parent with the best fitness to form a offspring *O1.*
Step 8	Repeat steps 6 and 7 until a new set of children is created which is of same size as the parent population
Step 9	Evaluate this new child set according to the kind of problem being solved.
Step 10	Utilize the incremental replacement method to create *P(t)* from the parent population and the set of children.

Ze stelden vast dat de heuristiek in de meeste gevallen in staat was om de optimale oplossing te vinden. Ze moesten echter constateren dat de computer extreem veel tijd in beslag nam om tot deze optimale oplossing te komen. In Jaramillo (1998) kunnen meer details gevonden worden.

3.4.4 Vergelijking van de metaheuristieken

Pirlot (1996) heeft in zijn onderzoek deze drie metaheuristieken vergeleken om zo de keuze van een metaheuristiek bij een welbepaald probleem te vergemakkelijken. Nochtans is het vergelijken van algoritmes niet eenvoudig is. Bovendien is het vrijwel zinloos om SA, TS en GA te vergelijken voor een bepaald optimalisatieprobleem zonder te specificeren welke implementaties gebruikt zijn.

In de meeste vergelijkingen tussen SA en TS werd TS in het algemeen superieur bevonden. De reden is voornamelijk dat TS in staat is om oplossingen van vergelijkbare kwaliteit te leveren in een veel kortere tijdspanne.

De conclusie die hierna gegeven wordt, wordt eerder bekeken vanuit het oogpunt van iemand die een levensecht probleem moet behandelen en dit probeert op te lossen dan van een academische OO[8]- onderzoeker.

3.5 Conclusie

In dit hoofdstuk zijn een aantal methoden aangehaald die oplossingen geven voor ons vestigingsmodel.

Wanneer we voor een gegeven probleem staan, vragen we ons best af of goede specifieke heuristieken of exacte methodes voor het probleem of varianten of subproblemen bestaan. Als we uiteindelijk een algemene heuristiek willen implementeren (eventueel bovenop een traditionele *local search*-heuristiek) is Pirlot (1996) voorstander om te beginnen met een eenvoudige heuristiek zoals SA of een basis KT[9]-versie van TS. Indien bemoedigende resultaten verkregen worden en betere resultaten nodig zijn, kunnen meer gesofisticeerde benaderingen gebruikt worden, zoals langere termijn-strategieën van TS. De tijd om SA of een eenvoudige TS te implementeren is inderdaad erg kort. Om een keuze te maken tussen SA en TS kan aangehaald worden dat TS over het algemeen sneller werkt. Het is vaak mogelijk om bij beide technieken oplossingen te verkrijgen van dezelfde kwaliteit. Wel moet vermeld worden dat zelfs een eenvoudige TS meer tactische keuzes vereist en daarom een beetje meer tijd nodig heeft om geïmplementeerd te worden.

HYBRIDE METAHEURISTIEKEN

Sommige heuristische algoritmen zijn eigenlijk een hybride van verschillende types metaheuristieken. Ze combineren elkaars betere kenmerken. Korte termijn-tabu search bijvoorbeeld is zeer goed in het vinden van lokale optima maar niet zo goed in het grondig verkennen van de verschillende delen van een toelaatbaar gebied om het deel te vinden dat het globaal optimum bevat. Doordat genetische algoritmen tegengestelde kenmerken hebben, kan

[8] Operationeel Onderzoek
[9] Korte Termijn

een verbeterd algoritme verkregen worden door deze twee te combineren. Zo wordt begonnen met een genetisch algoritme om de grootste hellingen te zoeken en dan, naar het einde toe, op een basisversie van *tabu search* overschakelen om snel van deze hellingen af te dalen. (Hillier & Lieberman, 2005, p. 653)

In het volgende hoofdstuk bespreken we de memetische algoritmes. Memetische algoritmen kunnen we ook in de categorie van hybride metaheuristieken plaatsen. Een hoofdkenmerk waarover de meeste MA-implementaties beschikken, is het gebruik van een populatiegebaseerde zoektocht die alle beschikbare kennis over het probleem probeert te gebruiken. Deze kennis zit in het MA vervat onder de vorm van heuristieken, lokale zoektechnieken en vele anderen. In deze thesis zal het MA gebaseerd zijn op een combinatie van GA en LS (*local search*).

4. Memetische algoritmen

Voordat we dieper ingaan op de memetische algoritmen, worden de evolutionaire algoritmen kort besproken. Vervolgens worden de memetische algoritmen uitgelegd en wordt getoond welk memetisch algoritme geïmplementeerd zal worden voor het onderzoek.

4.1 Evolutionaire algoritmen

Het idee om de Darwiniaanse principes toe te passen op geautomatiseerde oplossingsmethodes dateert vanaf 1948, lang voor de doorbraak van de computer. Methodes zoals *evolutionary programming*, *genetic algorithm*, *evolution strategies* worden sinds 1990 gezien als verschillende vertegenwoordigers van een algemene technologie die bekend is geworden onder *evolutionary computing*. Kort hierop wordt een vierde methodologie ontwikkeld die dezelfde ideeën volgt als zijn voorgangers: *genetic programming*. De algoritmes die betrokken zijn bij *evolutionary computing,* de evolutionaire algoritmes, zijn onder andere de genetische algoritmes die beschreven zijn in het vorige hoofdstuk. (Hart et al., 2005, p. 5)

Het algemeen idee achter de verschillende evolutionaire algoritmes is hetzelfde. Vertrekkend van een populatie worden mechanismen gebruikt om individuen met een hoge fitheid[10] te genereren. De mechanismen zijn gebaseerd op natuurlijke selectie en genetische variatie. Het algemene schema van een evolutionair algoritme in pseudocode kan als volgt weergegeven worden:

[10] Indien het een maximum-probleem is, is de fitheid een zo hoog mogelijke waarde. Indien het een minimum-probleem betreft, is de fitheid een zo laag mogelijke waarde.

Begin

INITIALISEER populatie met willekeurige kandidaat-oplossingen;

EVALUEER elke kandidaat;

Herhaal totdat (VOORWAARDE VOOR BEEINDIGING voldaan is) **Doe**

1. SELECTEER ouders;
2. HERCOMBINEER koppels van ouders;
3. MUTEER het resulterend nageslacht;
4. EVALUEER nieuwe kandidaten;
5. SELECTEER individuen voor de volgende generatie,

eindeDoe

Einde.

Figuur 4.1: Algemeen schema van een evolutionair algoritme (Hart et al., 2005, p. 6)

4.2 Memetische algoritmen: definitie en implementatie

Memetische algoritmen zijn een klasse van stochastische globale zoekheuristieken waarin benaderingen gebaseerd op evolutionaire algoritmen gecombineerd zijn met lokale zoektechnieken om de kwaliteit van de oplossingen, gecreëerd door de evolutie, te verbeteren (Hart et al., 2005, p.3). Ze behoren tot de klasse van hybride metaheuristieken (zie *supra*) aangezien ze de genetische algoritmen combineren met *local search*.

Memetische algoritmen zijn bijgevolg soortgelijk aan genetische algoritmen. Waar genetische algoritmen op het concept van biologische evolutie vertrouwen, vertrouwen memetische algoritmen op mimische culturele evolutie. Terwijl in de natuur de genen gewoonlijk niet wijzigen gedurende de levensduur van een individu, wijzigen de *memes* wel. (Digalakis en Margaritis, 2004)

De naam van memetische algoritmen vindt de oorsprong met de introductie van het woord '*meme*' uit het boek van R. Dawkins, 'The Selfish Gene'[11]:

[11] Dawkins, R. (1990) *The Selfish Gene*, Oxford University Press.

'*Examples of memes are tunes, ideas, catch-phrases, clothes fashions, ways of making pots or of building arches. Just as genes propagate themselves in the gene pool by leaping from body to body via sperms or eggs, so memes propagate themselves in the meme pool by leaping from brain to brain via a process which, in the broad sense, can be called imitation*.' (Moscato, 1999, p. 220) *Memes* kunnen gezien worden als eenheden van 'culturele overbrenging' op dezelfde manier als genen de eenheden van biologische overbrenging zijn.

Zoals eerder vermeld, maken memetische algoritmen gebruik van *local search* om de zoektocht naar de optimale oplossing te verbeteren. *Local search* is een zoekmethode die herhaaldelijk een verzameling van punten in een buurt van de huidige oplossing onderzoekt. De huidige oplossing wordt vervangen door een betere buur indien die bestaat. In de pseudocode voor een eenvoudig memetisch algoritme wordt de *local search* als volgt toegevoegd aan een genetisch algoritme:

```
Begin
    INITIALISEER populatie;
    EVALUEER elke kandidaat;
    Herhaal totdat ( VOORWAARDE VOOR BEEINDIGING voldaan is) Doe
        SELECTEER ouders;
        HERCOMBINEER om nageslacht te produceren;
        EVALUEER nageslacht;
        VERBETER nageslacht via Local Search;
        MUTEER nageslacht;
        EVALUEER nageslacht;
        VERBETER nageslacht via Local Search;
        SELECTEER individuen voor de volgende generatie;
    eindeDoe
Einde.
```

Figuur 4.2: Schema voor een eenvoudig memetisch algoritme met *local search* (Hart et al., 2005, p. 15)

De meest belangrijke factor in het ontwerp van een memetisch algoritme is waarschijnlijk de keuze van een verbeteringsheuristiek of een *local search move* operator. Dit laatste duidt op de manier waarop de verzameling van buurtpunten onderzocht wordt wanneer gezocht wordt naar een verbeterde oplossing. De keuze van *move* operator kan een dramatisch effect hebben op de efficiëntie en effectiviteit van de *local search* en dus ook het resulterend memetisch algoritme. (Hart et al., 2005, p. 16-17)

Memetische algoritmes kunnen last hebben van vroegtijdige convergentie, waarbij de populatie convergeert rond een suboptimaal punt. Indien de *local search* wordt toegepast tot elk punt beland is in een lokaal optimum, leidt dit tot een verlies aan diversiteit in de populatie tenzij nieuwe lokale minima constant geïdentificeerd worden. Dit kan verholpen worden door de *local search* enkel toe te passen op een kleine fractie van de populatie. Dit helpt te verzekeren dat de rest van de populatie verschillend blijft. (Hart et al., 2005, p. 16)

Het memetisch algoritme dat in het onderzoek van deze eindverhandeling gebruikt wordt, wordt gegeven in figuur 4.3.

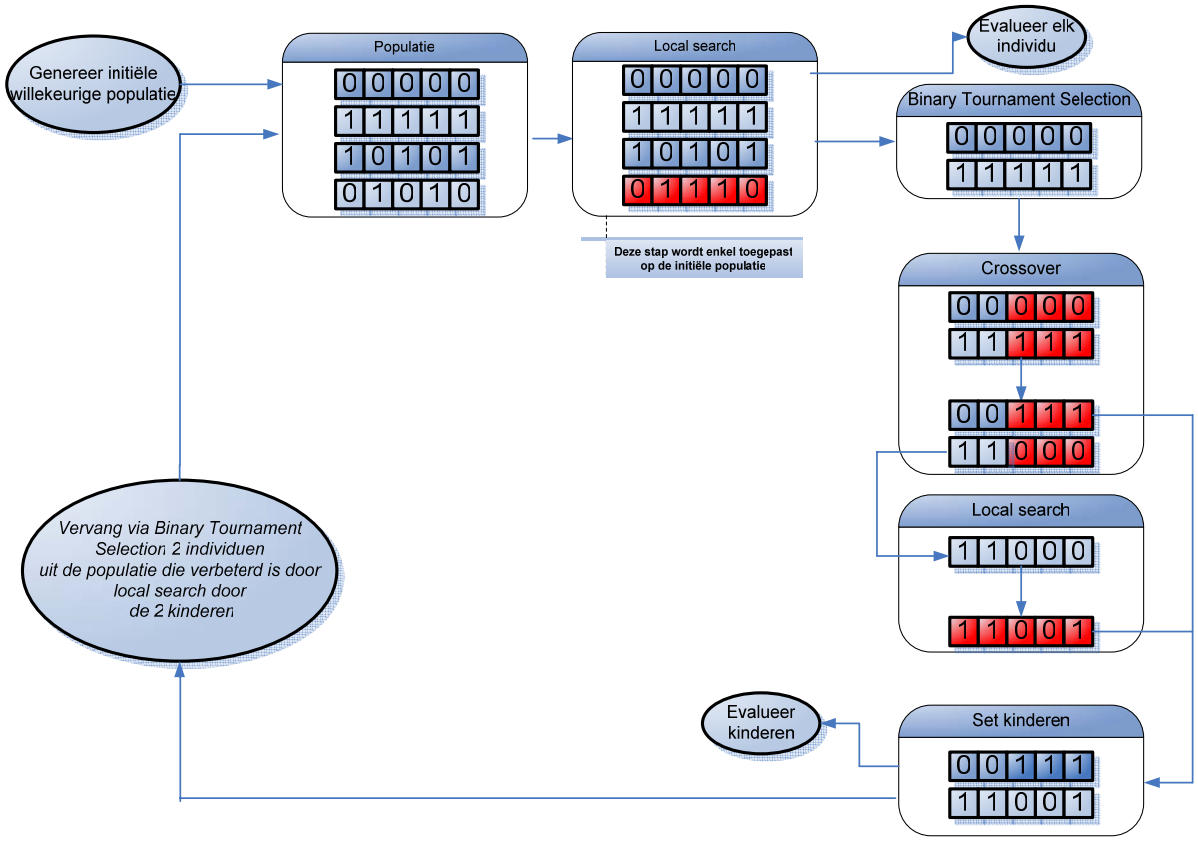
Begin
INITIALISEER populatie;
EVALUEER elke kandidaat;
VERBETER met een bepaalde kans de populatie via *Local Search*
Herhaal totdat (VOORWAARDE VOOR BEEINDIGING voldaan is) **Doe**
SELECTEER twee ouders via *Binary Tournament Selection*;
HERCOMBINEER om twee kinderen te produceren via O*ne-Point Crossover*;
EVALUEER twee kinderen;
VERBETER met een bepaalde kans de twee kinderen via *Local Search*;
SELECTEER twee individuen van de populatie via *Binary Tournament Selection* en vervang ze door de twee kinderen;
eindeDoe
Einde.

Figuur 4.3: Schema voor het memetisch algoritme van deze eindverhandeling

Het volgende hoofdstuk verschaft meer informatie over hoe het memetisch algoritme precies geïmplementeerd wordt in het informatiesysteem voor het onderzoek. In dit deel wordt nog ingegaan op *Binary Tournament Selection*.

De *Binary Tournament Selection*-methode kiest *at random* twee individuen, waarvan het individu met de hoogste fitheid wordt geselecteerd om kinderen te produceren. Deze methode kan zeer efficiënt geïmplementeerd worden. Bovendien is bewezen dat deze methode oplossingen van betere kwaliteit geeft in vergelijking met andere methodes (Jaramillo et al., 2002). In deze eindverhandeling wordt gebruik gemaakt van deze selectiemethode.

Tot slot verschaft figuur 4.5 een duidelijk beeld over de werking van het memetisch algoritme dat in dit onderzoek geïmplementeerd wordt.



Figuur 4.4: De structuur van het memetisch algoritme voor deze eindverhandeling

5. Het onderzoek

Voor we tot het onderzoek overgaan, wordt het gebruikte informatiesysteem uitgelegd. Geregeld worden voorbeelden voorzien. In deel twee wordt duidelijk gemaakt welke parameters we kunnen onderscheiden voor ons onderzoek en in welke mate de parameters zullen variëren. Deel drie toont aan hoe de datasets gegenereerd worden.

5.1 Het gebruikte informatiesysteem

De software die gebruikt werd is een combinatie van reeds bestaande software. Het programma is geschreven in de programmeertaal 'PASCAL' en gecompileerd tot een uitvoerbaar programma voor het Linux besturingssysteem. De broncode is terug te vinden in bijlage I.

Inputfile

2	3
10	250
15	300
5	
12	3
6	
11	8
3	
6	14

Kleurenindex:	
	vestiging 1
	vestiging 2
	klant 1
	klant 2
	klant 3

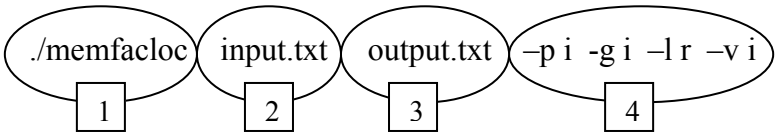
Tabel 5.1: Voorbeeld van een dataset als inputfile voor het memetisch algoritme

De twee getallen op de eerste rij, namelijk 2 en 3, geven respectievelijk het aantal potentiële vestigingen en het aantal klanten weer. De getallen in de twee daaropvolgende rijen geven per vestiging de capaciteit en de kost om de vestiging te plaatsen weer. Vervolgens worden per klant twee rijen voorzien. Op de eerste rij staat de totale vraag van de klant; de tweede rij geeft per vestiging de kost weer om één eenheid te transporteren van potentiële vestiging 1 en

2 naar deze klant. De dataset mag enkel gehele getallen bevatten. Indien niet gehele waarden zijn opgenomen in de dataset, worden deze naar onder afgerond.

Commando

Het commando om het memetisch algoritme uit te voeren bestaat uit vier delen:



Het eerste deel, ./memfacloc, zorgt dat het programma van het memetisch algoritme wordt uitgevoerd. De dataset wordt gekozen in het tweede deel, bijvoorbeeld data/miclus.txt. In het derde deel wordt de naam van de outputfile gedefinieerd, bijvoorbeeld test1.txt. Het vierde deel tenslotte biedt de mogelijkheid om over een aantal factoren te beslissen, de zogenaamde parameters: het aantal generaties, de populatiegrootte, de kans dat *local search* wordt uitgevoerd en de oplossingsweergave (zie tabel 5.1). Standaard bevat het programma voor p, g, l en v respectievelijk de waarden 20; 0,2; 10 en 1.

Tabel 5.2: De parameters van het memetisch algoritme

Parameters	Mogelijke waarden	Betekenis
-p i	$i \in N_0$	Aantal individuen in de populatie
-g i	$i \in N_0$	Aantal generaties
-l r	r = [0,1]	Kans dat *local search* wordt uitgevoerd
-v i	i = {0, 1, 2, 3}	Graad van oplossingsweergave

De oplossing kan in vier verschillende graden worden weergegeven. Graad 0 geeft enkel de beste oplossing weer. De volgende graad laat ook de initiële populatie en alle oplossingen die toegevoegd zijn aan de populatie zien. Graad 2 onthult hiernaast veel meer over de werking van het algoritme, zoals de *crossover* en de *local search* die uitgevoerd werden. De derde graad tenslotte vult graad 2 aan met het tonen van elke gegenereerde populatie.

Werking van het algoritme

- Kostenberekening

De doelfunctie minimaliseert de totale kost van de gekozen oplossing. Deze totale kost omvat de vaste kosten van de geopende vestigingen en de transportkosten (zie het wiskundig model in 2.2). Het transportalgoritme van Ford Fulkerson wordt gebruikt om de transportkosten te minimaliseren. Mits het volledig uitleggen van dit algoritme buiten het bestek van deze eindverhandeling ligt, wordt het algoritme kort samengevat en wordt verder verwezen naar (Winston, 1994, p. 405-412):

Stap 1 Vind een toelaatbare stroom.
Stap 2 Probeer aan de hand van de *labeling*[12] procedure de eindbestemming te *labelen*. Indien de eindbestemming niet *gelabeld* kan worden is de huidige toelaatbare stroom een maximumstroom; indien de eindbestemming niet *gelabeld* is, ga naar Stap 3.
Stap 3 Pas de toelaatbare stroom aan en verhoog de stroom van de bron naar de eindbestemming via de methode beschreven in (Winston, 1994, p. 407-408). Ga terug naar Stap 2.

- Stappen van het algoritme

Om te beginnen wordt *at random* een populatie gekozen: elke vestiging heeft 50% kans om geopend te worden. Stel dat we over vijf potentiële vestigingen beschikken, dan wordt een oplossing in de populatie als volgt weergegeven:

1 2 3	6958

Dit wil zeggen dat de potentiële vestigingen 1, 2 en 3 geopend zijn en vestigingen 4 en 5 gesloten zijn. De totale kost bedraagt 6958.

[12] *Labeling* is het 'etiketteren' van het netwerk. Op deze manier kan een stroom door het netwerk gevonden worden om de totale *flow* te verhogen.

Bij elke generatie worden stochastisch twee maal twee oplossingen uit de populatie gehaald. Van deze twee koppels wordt aan de hand van de doelfunctie de beste oplossing eruit gehaald, zodat men de twee overgebleven partners kan koppelen. Hier wordt een genetische *crossover* op toegepast waarbij het *crossover*-punt willekeurig bepaald wordt. Het memetisch algoritme bevat geen mutatiemechanisme, maar *local search* kan toegepast worden aan een vooraf bepaalde kans. Laten we de vorige oplossing binair formuleren (eerste regel tabel):

1	1	1	0	0	6958
0	1	1	0	0	11040
1	0	1	0	0	8924
1	1	0	0	0	7789
1	*1*	*1*	*1*	*0*	*6690*
1	1	1	0	1	12980

Local search gaat bij elke stap één vestiging openen en terug sluiten van de huidige oplossing (1, 1, 1, 0, 0). Aangezien we vijf potentiële vestigingen hebben, resulteert *local search* in vijf nieuwe oplossingen. De totale kost staat achter de nieuwe oplossingen. In dit geval ligt de totale kost lager dan die van de huidige oplossing als we vestigingen 1, 2, 3 en 4 openen. Deze oplossing wordt bijgevolg geselecteerd als nieuwe huidige oplossing. Vanuit deze nieuwe huidige oplossing wordt opnieuw *local search* gedaan. Indien *local search* geen betere oplossingen dan de huidige oplossing biedt, wordt de huidige oplossing in de populatie geplaatst. Indien dit niet zo is, wordt vanuit de nieuwe huidige oplossing opnieuw *local search* gedaan.

De twee verbeterde oplossingen zullen twee slechtere oplossingen van de populatie vervangen. Deze slechtere oplossingen worden uit de populatie gehaald via hetzelfde selectiemechanisme: twee maal twee oplossingen worden geselecteerd. Van elk koppel wordt vervolgens de slechtste oplossing vervangen door een verbeterde oplossing.

De capaciteitsbeperking wordt geïmplementeerd door een extra kost aan ontoelaatbare oplossingen toe te voegen. Indien de totale vraag de totale capaciteit van de geopende vestigingen overstijgt heeft de deze ontoelaatbare oplossing als totale kost ‘Int64Inf –

TotalCapacity'[13]. Daardoor zijn toelaatbare oplossingen steeds beter als ontoelaatbare oplossingen. De vaste kosten en variabele kosten mogen niet van dezelfde grootteorde zijn als Int64Inf.

Tenslotte mag niet vergeten worden dat de vraag van een klant kan toegewezen worden aan verschillende vestigingen. Dit is echter niet terug te vinden in de oplossing aangezien wij ons concentreren op de vraag: 'Welke vestigingen worden geopend en welke niet?'

5.2 Experimentele opzet

In het onderzoek wordt gebruik gemaakt van een aantal parameters: ruimtelijke parameters, klantgebonden parameters, vestigingsparameters, kostparameters en sturingsparameters. Elke parameter zal kort toegelicht worden. Voor de sturingsparameters wordt hier reeds aangegeven welke waarden zij kunnen aannemen. De overige parameters worden besproken in deel 5.3.

Ruimtelijke parameters

Hierbij wordt de vraag gesteld waar en hoe de klanten gelegen zijn; willekeurig verspreid of in clusters. Ook wordt bepaald hoeveel klanten en hoeveel potentiële vestigingen er zijn.

We kiezen om datasets van drie verschillende groottes te onderzoeken, namelijk een kleine onderneming, een middelgrote onderneming en een grote onderneming. De kleine onderneming telt 25 klanten en 12 potentiële vestigingen, de middelgrote telt 100 klanten en 40 potentiële vestigingen en de grote onderneming tenslotte telt 500 klanten en 50 potentiële vestigingen. Per grootte onderneming worden twee datasets gemaakt waardoor we aan 6 datasets komen. In de eerste reeks datasets liggen de in clusters liggen en in de tweede liggen ze willekeurig verspreid. De potentiële vestigingen behouden voor eenzelfde grootte onderneming dezelfde locaties.

[13] Int64Inf is het grootst mogelijke getal dat in het dataype Int64 past (een 64-bit integer type). Hiervan wordt de totale capaciteit van de betreffende oplossing afgetrokken.

Dit geeft:

- Kleine onderneming met 25 klanten en 12 potentiële vestigingen
    - Klanten liggen willekeurig verspreid
    - Klanten liggen geclusterd
- Middelgrote onderneming met 100 klanten en 40 potentiële vestigingen
    - Klanten liggen willekeurig verspreid
    - Klanten liggen geclusterd
- Grote onderneming met 500 klanten en 50 potentiële vestigingen
    - Klanten liggen willekeurig verspreid
    - Klanten liggen geclusterd

Klantgebonden parameters

Deze parameters houden de vraag van de klanten in. Deze parameter wordt gedurende het onderzoek niet gewijzigd. Bij aanvang wordt voor de klanten van de drie verschillende groottes ondernemingen de vraag bepaald. De vraag is per grootte onderneming hetzelfde, ongeacht de klanten geclusterd zijn of willekeurig verspreid liggen.

Vestigingsparameters

De vestigingen beschikken elk over een bepaalde capaciteit. Per ondernemingsgrootte worden de capaciteiten voor de verschillende vestigingen vastgelegd. De capaciteiten kunnen per ondernemingsgrootte variëren of zijn identiek voor elke vestiging. Zo komen we aan 12 datasets. De datasets waarbij de vestigingen per dataset identieke capaciteiten hebben zorgen voor een gemakkelijke vergelijkingsbasis voor verschillende aspecten. Aan de andere kant hebben we de datasets met variërende capaciteiten. Niet alle ziekenhuizen, opslagplaatsen, fastfood-restaurants zijn immers even groot.

Samengevat hebben we:

- Kleine onderneming met 25 klanten
  - 12 potentiële vestigingen met identieke capaciteiten
    - Klanten liggen willekeurig verspreid
    - Klanten liggen geclusterd
  - 12 potentiële vestigingen met variërende capaciteiten
    - Klanten liggen willekeurig verspreid
    - Klanten liggen geclusterd
- Middelgrote onderneming met 100 klanten en 40 potentiële vestigingen met dezelfde capaciteit
  - 40 potentiële vestigingen met identieke capaciteiten
    - Klanten liggen willekeurig verspreid
    - Klanten liggen geclusterd
  - 40 potentiële vestigingen met variërende capaciteiten
    - Klanten liggen willekeurig verspreid
    - Klanten liggen geclusterd
- Grote onderneming met 500 klanten en 50 potentiële vestigingen met dezelfde capaciteit
  - 50 potentiële vestigingen met identieke capaciteiten
    - Klanten liggen willekeurig verspreid
    - Klanten liggen geclusterd
  - 50 potentiële vestigingen met variërende capaciteiten
    - Klanten liggen willekeurig verspreid
    - Klanten liggen geclusterd

Kostparameters

De totale kost bestaat uit twee componenten. De eerste component, namelijk de vaste kost, hangt af van de capaciteit van een vestiging. De variabele kost wordt berekend door de euclidische afstand tussen elke klant en elke potentiële vestiging te vermenigvuldigen met een

kost/km per eenheid en het aantal eenheden dat van de betreffende vestiging naar die klant getransporteerd moeten worden. De kostparameters wijzigen niet gedurende het onderzoek.

Sturingsparameters

De sturingsparameters zijn de parameters van het programma van het memetisch algoritme, uitgezonderd 'de graad van de oplossingsweergave' (zie deel 5.1). Hier wordt mee gespeeld door dit in het vierde deel van het commando (zie deel 5.1) in te geven. Wijzigingen in één van de vorige parameters resulteert in een nieuwe dataset die als *inputfile* in het tweede deel van het commando moet worden ingegeven.

De sturingsparameters kunnen elk twee waarden aannemen in het onderzoek. Dit resulteert in acht combinaties. Elke parameter kan de standaardwaarde in het memetisch algoritme aannemen. Vervolgens dient een tweede reeks waarden bepaald te worden. Aan de hand van enkele testen worden volgende waarden gekozen voor de populatiegrootte, het aantal generaties en de kans op *local search*: 40, 40 en 80%.

5.3 Het genereren van de datasets

Echte regels voor het genereren van datasets bestaan niet. De manier waarop de datasets voor deze thesis gegenereerd worden, berust eerder op een gevoel of pure logica.

Alle gegevens worden gegenereerd in Microsoft Excel. Veelal wordt van de 'ASELECTTUSSEN'-functie gebruik gemaakt om willekeurige gehele getallen te generen die tussen twee bepaalde waarden liggen.

Assenstelsel

Het assenstelsel varieert voor de drie ondernemingengroottes. Zo wordt gesimuleerd dat grotere ondernemingen niet alleen meer klanten en vestigingen hebben, maar ook een groter

grondgebied beslaan. De reikwijdte van het assenstelsel voor de kleine, middelgrote en grote onderneming bedraagt respectievelijk 100-100, 140-140 en 250-250.

Vraag van de klanten

Voor de vraag van de klanten van alle datasets worden willekeurige getallen gegenereerd tussen 1 en 15. Dit resulteert in drie verschillende reeksen gegevens: vraag_klein, vraag_middel en vraag_groot. De gegevens blijven dezelfde ongeacht de klanten willekeurig verspreid liggen of geclusterd zijn.

Ligging van de klanten

Hier wordt een onderscheid gemaakt naargelang de klanten in clusters gelegen zijn of niet.

Klanten zijn willekeurig verspreid

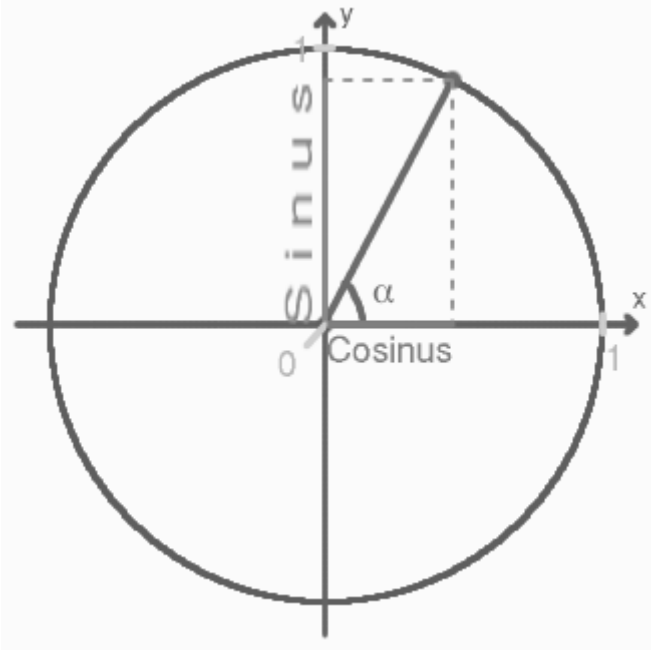
Als de klanten willekeurig verspreid over heel het oppervlakte van het assenstelsel liggen, worden de x- en y-coördinaat beiden bepaald door een willekeurig getal te nemen tussen 0 en de reikwijdte van het assenstelsel.

Klanten zijn gelegen in clusters

In dit geval verloopt het bepalen van de coördinaten wat complexer. Eerst wordt *at random* het aantal clusters bepaald. Dit is een getal tussen 2 en 6 voor de kleine onderneming en tussen 3 en 10 voor de middelgrote en grote onderneming. De reden voor dit verschil is dat de kleine onderneming slechts 25 klanten heeft. Zo liggen de klanten van de kleine onderneming in 4 clusters, die van de middelgrote onderneming in 5 clusters en die van de grote onderneming in 9 clusters. Dan wordt per cluster een middelpunt vastgelegd. Dit gebeurt zoals de bepaling van de klanten die willekeurig verspreid zijn. Hierna moet beslist worden hoeveel klanten zich in elke cluster bevinden. Dit gebeurt door per cluster een willekeurig getal in een interval te genereren. De grenswaarden van het interval hangen af van het aantal clusters en het aantal klanten. Vervolgens wordt per cluster een straal vastgelegd. Voor de kleine onderneming ligt de lengte van de straal tussen 5-15, voor de middelgrote onderneming

tussen 8-25 en voor de grote onderneming tussen 8-18. De achterliggende reden voor deze keuzes ligt in het aantal clusters waarin de klanten liggen en de reikwijdte van het assenstelsel.

Eens dit allemaal vastligt, kan begonnen worden aan de bepaling van de x- en y-coördinaten van de klanten per cluster. Eerst wordt per klant de afstand bepaald hoever hij/zij van het middelpunt van de cluster gelegen is. Dit gebeurt logischerwijze door een willekeurig getal te genereren tussen 0 en de lengte van de straal. Vervolgens dient de hoek α bepaald te worden die de klant met een ingebeelde horizontale rechte door het middelpunt maakt. Dit leidt tot de keuze van een willekeurig getal tussen 0 en 360 (graden). Vervolgens worden met behulp van de goniometrie de coördinaten bepaald (zie figuur 5.1).



Figuur 5.1: Goniometrische cirkel

Eerst worden de cosinus en de sinus berekend aan de hand van de graden die per klant vastgelegd zijn. Dit kunnen zowel positieve als negatieve getallen zijn. Dan worden deze uitkomsten vermenigvuldigd met de afstand van de klant en afgerond naar het dichtstbijzijnde geheel getal. Zo wordt een koppel coördinaten verkregen. Tot slot dienen dit coördinaat, dat

momenteel in een cluster rond het punt (0,0) gelegen is, opgeteld en/of afgetrokken te worden bij/van het middelpunt van de cluster.

De gegevens betreffende de ligging en de vraag van de klanten zijn terug te vinden in bijlage IV.

Capaciteiten van de potentiële vestigingen

Eerst en vooral moet de totale klantenvraag gekend zijn omdat de totale klantenvraag de totale capaciteit niet mag overschrijden. Ook is het nefast als de capaciteit met een zeer grote waarde de totale klantenvraag zou overschrijden. Het probleem zou dan immers omslaan in een *uncapacitated* probleem.

Ook hier moet een onderscheid gemaakt worden tussen de datasets waar de vestigingen variërende capaciteiten hebben en de datasets die vestigingen met identieke capaciteiten bevatten.

Variërende capaciteiten

In dit geval worden wederom willekeurige getallen gegenereerd uit een interval. Voor de kleine onderneming is dit interval [20, 40], voor de middelgrote onderneming [40, 80] en voor de grote onderneming [80, 150].

Identieke capaciteiten

Nadat de totale vraag van de klanten is berekend, wordt aan de hand van het aantal potentiële vestigingen de capaciteit bepaald. Voor de kleine, middelgrote en grote onderneming geeft dit respectievelijk 30, 50 en 110.

Vaste kost van de potentiële vestigingen

Het openen van een vestiging houdt een eenmalige vaste kost in. Mits de vaste kost niet onafhankelijk is van de capaciteit van een vestiging, wordt deze voor de twee categorieën als volgt bepaald.

Variërende capaciteiten

De capaciteit wordt vermenigvuldigd met een willekeurig getal tussen 0.8 en 1.3. Dit willekeurig getal kan voor elke vestiging verschillend zijn. Om de kost van de potentiële vestiging te verkrijgen, wordt de uitkomst vermenigvuldigen met 100.

Identieke capaciteiten

Met het oog op het verkrijgen van een eenvoudige vergelijkingsbasis, worden de capaciteiten simpelweg vermenigvuldigd met 100.

Ligging van de potentiële vestigingen

De bepaling van de ligging van de potentiële vestigingen gebeurt op identieke wijze als de bepaling van de ligging van de klanten die willekeurig verspreid liggen.

De gegevens voor dit onderzoek betreffende de potentiële vestigingen, namelijk ligging, vaste kost en capaciteit, zijn terug te vinden in bijlage V.

Variabele kost

De variabele kost wordt berekend door de euclidische afstand tussen elke klant en elke potentiële vestiging te vermenigvuldigen met een kost/km per eenheid en het aantal eenheden dat van de betreffende vestiging naar die klant getransporteerd moeten worden (zie *supra*). De kost/km per eenheid wordt vastgelegd op 30.

Nadat al deze gegevens vastliggen, kunnen met behulp van het programma in bijlage II de datasets gegenereerd worden. Deze datasets zijn gebruiksklaar als *inputfile* voor het programma van het memetisch algoritme.

6. Analyse van de resultaten

Eerst wordt *factorial design* uitgelegd. Via deze methode kunnen onze resultaten geanalyseerd worden. Vervolgens wordt dit toegepast op de resultaten van het onderzoek. Tot slot worden conclusies getrokken uit de voorgaande analyse.

6.1 Methode voor de analyse

Law en Kelton (2000) bespreken in hun boek *factorial design*. Dit is een methode om resultaten te analyseren in een onderzoek waar een onderscheid gemaakt kan worden in factoren en antwoorden. De factoren, in ons onderzoek de sturingsparameters (zie 5.2), kunnen elk twee waarden aannemen. Aan de laagste waarde wordt een – toegekend, aan de hoogste een +. Concreet ziet dit voor het onderzoek als volgt uit:

Tabel 6.1: De waarden van de factoren

Factor	-	+
p	20	40
g	10	40
l	0.2	0.8

De min-kolom bevat de standaardwaarden van het memetisch algoritme. De waarden van de plus-kolom liggen allemaal hoger. Deze keuze berust op een aantal uitgevoerde testen met de datasets waaruit bleek dat het niet nodig was om nog hogere waarden te kiezen.

De antwoorden zijn de resultaten van het memetisch algoritme, namelijk ‘de beste gevonden oplossing’. Elke factorcombinatie heeft één antwoord (Law & Kelton, 2000, p. 660):

Tabel 6.2: Matrix voor een 2^3 *factorial design*

Factorcombinatie	Factor 1	Factor 2	Factor 3	Antwoord
1	-	-	-	A1
2	+	-	-	A2
3	-	+	-	A3
4	+	+	-	A4
5	-	-	+	A5
6	+	-	+	A6
7	-	+	+	A7
8	+	+	+	A8

Voor elk van de 12 datasets (zie *supra*) zal dit toegepast worden zodat de verkregen resultaten geanalyseerd en vergeleken kunnen worden. Eerst wordt het **hoofdeffect** van elke factor geanalyseerd. Vervolgens wordt het **interactie-effect** van telkens twee factoren berekend.

Het **hoofdeffect** van factor *j* is de gemiddelde verandering in het antwoord die veroorzaakt wordt door factor *j* te verplaatsen van het "-" niveau naar het "+" niveau terwijl alle andere factoren constant gehouden worden. Het hoofdeffect van factor 1:

$$e_1 = \frac{(A_2 - A_1) + (A_4 - A_3) + (A_6 - A_5) + (A_8 - A_7)}{4}$$

waarbij 4 gelijk is aan 2^{3-1}. De berekeningen voor e_2 en e_3 verlopen analoog. Het hoofdeffect meet de gemiddelde verandering in het antwoord veroorzaakt door een verandering in een individuele factor. Het is mogelijk dat het effect van factor j_1 op een bepaalde manier van het niveau van factor j_2 afhangt. In dit geval wordt gesproken van een interactie tussen beide factoren. (Law & Kelton, 2000, p.661)

Het **interactie-effect** wordt gedefinieerd als de helft van het verschil tussen het gemiddelde effect van factor j_1 wanneer factor j_2 vastgesteld is op het "+" niveau en het gemiddelde effect van j_1 wanneer j_2 vastgesteld is op het "-" niveau. De overige factoren (hier: factor) worden constant gehouden. Het interactie-effect tussen factor 1 en 2 wordt als volgt berekend:

$$e_{12} = \frac{1}{2}\left[\frac{(A_4 - A_3) + (A_8 - A_7)}{2} - \frac{(A_2 - A_1) + (A_6 - A_5)}{2}\right]$$

Hoe moeten deze resultaten geïnterpreteerd worden? Het hoofdeffect van de eerste factor kan als volgt begrepen worden. Het gemiddelde effect om factor 1 van het "-" niveau naar het "+" niveau te brengen is de stijging/daling van de totale kost van het vestigingsprobleem met e_1. Indien het interactie-effect in de buurt van 0 komt, wijst dit op een minimale interactie tussen beide factoren. Zoniet, beïnvloeden de factoren elkaar positief of negatief. In dit geval bestaat voor *factorial design* met drie factoren de mogelijkheid om een grafiek op te stellen om zo het interactie-effect duidelijk af te lezen. Enkele bijkomende berekeningen dienen gemaakt te worden. Hiervoor wordt verwezen naar de literatuur (Cobb, 1998, p. 392-396).

Opstellen van een betrouwbaarheidsinterval voor de resultaten

Aangezien de A_i's willekeurige variabelen zijn, zijn de effecten ook willekeurig. Om uit te zoeken of de effecten "echt" zijn dient hun variantie geschat te worden. Door n keer het hele ontwerp te herhalen, worden n onafhankelijke waarden van elk effect verkregen. Deze kunnen dan gebruikt worden om betrouwbaarheidsintervallen van bij benadering $100(1-\alpha)\%$ te vormen voor de verwachte effecten. Voor dit onderzoek worden voor elke dataset vijf replicaties uitgevoerd ($n = 5$). (Law & Kelton, 2000, p. 664)

We hebben hier te maken met een kleine steekproef aangezien $n < 30$. De steekproefverdeling van het gemiddelde hangt af van de kansverdeling van de populatie. We nemen aan dat de populatie normaal verdeeld is. Hierdoor kan gezegd worden dat de steekproefverdeling van het gemiddelde ook een normale verdeling is. Mits de populatiestandaarddeviatie voor ons onderzoek niet bekend is, wordt de steekproefstandaarddeviatie gebruikt om deze te schatten. Deze wordt voor het eerste hoofdeffect als volgt berekend:

$$s_1 = \sqrt{\frac{\sum_{i=1}^{n}(e_{1i} - \bar{e}_1)^2}{n-1}}$$

Waarbij $\bar{e}_1$ het gemiddelde van de vijf replicaties is. Voor de overige twee hoofdeffecten en de drie interactie-effecten verloopt de berekening analoog. (Anderson et al., 1997, p. 211)

Om te zien of een effect statistisch significant is, wordt aan de hand van de vijf replicaties een betrouwbaarheidsinterval opgesteld. De betekenis van een betrouwbaarheidsinterval is als volgt: indien voor 100 steekproeven van grootte *n* uit een populatie bijvoorbeeld een 95%-betrouwbaarheidsinterval geconstrueerd wordt, mag verwacht worden dat 95 van de 100 intervallen het echte gemiddelde bevat en dat 5 intervallen het echte gemiddelde niet bevatten. We kunnen met andere woorden met 95% zekerheid zeggen dat het interval het echte gemiddelde bevat. Indien nul deel uitmaakt van het betrouwbaarheidsinterval is het betreffende effect niet significant verschillend van nul. (Anderson et al., 1997, p. 205; Law & Kelton, 2000, p.664) In deze eindverhandeling worden betrouwbaarheidsintervallen van 95% en 99% gehanteerd.

Het geschikte betrouwbaarheidsinterval voor een kleine steekproef wordt gebaseerd op een kansverdeling die bekend staat als de *t*-verdeling. De *t*-verdeling omvat een parameter, namelijk de vrijheidsgraden. Deze vrijheidsgraden hebben betrekking op het aantal onafhankelijke hoeveelheden die in de berekening van de variantie opgenomen worden. Dit aantal bedraagt $n - 1$. Want als $n - 1$ van de $(e_i - e)$-waarden gekend zijn, kan de resterende waarde exact bepaald worden door gebruik te maken van de voorwaarde dat de som van de $(e_i - e)$-waarden nul moet zijn. (Anderson et al., 1997, p. 211)

6.2 Analyse

Eerst worden de datasets behandeld waarbij de capaciteiten van de potentiële vestigingen variëren. Daarna worden de datasets met identieke capaciteit geanalyseerd. De benaming bestaat uit drie delen. Het eerste deel duidt op de ondernemingsgrootte. Het tweede deel geeft

de ligging van de klanten weer. Of de potentiële vestigingen over variërende of identieke capaciteiten beschikken, wordt vermeld in het derde deel. De volledige resultaten zijn terug te vinden in bijlage III. In bijlage V wordt duidelijk welke vestigingen geopend worden aan de hand van tabellen en bijhorende grafieken.

POTENTIELE VESTIGINGEN MET VARIERENDE CAPACITEITEN

Klein-geclusterd-variërend

Uit de resultaten van het onderzoek blijkt dat een wijziging in de sturingsparameters geen enkel effect heeft op de 'beste oplossing'. Zowel wanneer de sturingsparameters op het "-" niveau liggen als wanneer ze op het "+" niveau liggen, is de gevonden beste oplossing gelijk aan 161 129. Dit is zo voor elke replicatie. Acht vestigingen worden geopend.

Klein-willekeurig-variërend

Ook hier komt elke keer dezelfde beste oplossing tevoorschijn. De totale kost is gelijk aan 142 881 waarbij tien vestigingen worden geopend.

Middel-geclusterd-variërend

Zoals in de twee voorgaande gevallen, hebben de sturingsparameterwaarden geen enkele invloed op de beste oplossing die verkregen wordt. De totale kost van de beste oplossing bedraagt 456 653. Veertien vestigingen worden geopend.

Middel-willekeurig-variërend

Hier vertoont het algoritme wel verschillende resultaten. Deze worden als volgt weergegeven:

Tabel 6.3: De resultaten van de dataset middel-willekeurig-variërend

	replicatie nr.				
factorcombinatie	1	2	3	4	5
1: p20g10l2	401195	401301	401245	401688	400180
2: p40g10l2	401245	400615	400180	400074	400074
3: p20g40l2	405813	403126	400180	400517	401195
4: p40g40l2	400074	402267	400074	400509	400074
5: p20g10l8	400074	400517	400074	400074	400509
6: p40g10l8	400074	400074	400074	400180	400074
7: p20g40l8	400074	400074	400074	400074	400074
8: p40g40l8	400074	400074	400074	400074	400074

Eerst worden de hoofd- en interactie-effecten per replicatie berekend. Dan wordt per hoofd- en interactie-effect het gemiddelde berekend en een betrouwbaarheidsinterval van 95% en van 99% opgesteld. De *t*-waarde bedraagt dan respectievelijk 2,776 en 4,604. De gemiddelden, samen met het betrouwbaarheidsinterval, per effect zijn:

Tabel 6.4: De betrouwbaarheidsintervallen voor de effecten van de dataset middel-willekeurig-variërend

		interval 95%		interval 99%	
	gemiddelde	beginwaarde	eindwaarde	beginwaarde	eindwaarde
hoofdeffect 1	-601,3	-1117,36114	-85,23885985	-1457,188145	254,5881445
hoofdeffect 2	252,4	-343,1006783	847,9006783	-735,238733	1240,038733
hoofdeffect 3	-958,1	-1862,52192	-53,67807955	-2458,085058	541,8850583
interactie-effect 1-2	-182	-996,5748778	632,5748778	-1532,973609	1168,973609
interactie-effect 1-3	524,1	-41,22933637	1089,429336	-413,499519	1461,699519
interactie-effect 2-3	-350,8	-990,9975847	289,3975847	-1412,568617	710,9686167

Enkel hoofdeffecten 1 en 3 zijn significant verschillend van nul op het 95%-niveau. We kunnen dus zeggen dat een populatiegrootte van 40 ten opzichte van een van 20 een beste oplossing levert die lagere totale kosten omvat. Ook het instellen van een kans op *local search* van 80% leidt tot betere resultaten. Op het 99%-niveau zijn hoofdeffecten 1 en 3 echter niet meer significant verschillend van nul. De overige effecten bevatten nul in het interval dus deze effecten zijn niet significant verschillend van nul.

De best verkregen oplossing opent 16 vestigingen en heeft een totale kost van 400 074.

Groot-geclusterd-variërend

De beste oplossing bedraagt in elke replicatie voor elke factorcombinatie 3 647 243 met 35 geopende vestigingen.

Groot-willekeurig-variërend

Bij de verspreide oplossing valt evenzeer één enkele beste oplossing waar te nemen. De totale kost voor deze oplossing bedraagt 2 888 069 waarbij 40 vestigingen geopend worden.

POTENTIELE VESTIGINGEN MET IDENTIEKE CAPACITEITEN

Hier kunnen analoge resultaten bemerkt worden als bij datasets met potentiële vestigingen met variërende capaciteiten.

Klein-geclusterd-identiek

Dezelfde beste oplossing komt wederom bij elke replicatie en bij elke factorcombinatie voor. De totale kost bedraagt 163 203 en acht vestigingen worden geopend.

Klein-willekeurig-identiek

De enige beste oplossing die te voorschijn komt opent tien vestigingen. De totale kost bedraagt 134 846.

Middel-geclusterd-identiek

De beste oplossing is overal dezelfde en bedraagt 490 493. Zestien vestigingen worden geopend.

Middel-willekeurig-identiek

Hier worden verschillende ‘beste oplossingen’ bekomen.

Tabel 6.5: De resultaten van de dataset middel-willekeurig-identiek

	replicatie nr.				
factorcombinatie	1	2	3	4	5
1: p20g10l2	381170	380603	382074	380596	380603
2: p40g10l2	382074	382456	380603	382456	382074
3: p20g40l2	382648	380596	382074	384407	382074
4: p40g40l2	380596	380596	382074	380596	380596
5: p20g10l8	380596	380596	380596	380596	380596
6: p40g10l8	380596	380596	380596	380596	380596
7: p20g40l8	380596	380596	380596	380596	380596
8: p40g40l8	380596	380596	380596	380596	380596

Analoog aan de 'variërende' versie worden de gemiddelden en de begin- en eindwaarden van het interval als volgt weergegeven:

Tabel 6.6: De betrouwbaarheidsintervallen voor de effecten van de dataset middel-willekeurig-identiek

		interval 95%		interval 99%	
	gemiddelde	beginwaarde	eindwaarde	beginwaarde	Eindwaarde
hoofdeffect 1	-136,2	-558,0645287	285,6645287	-835,8629287	563,4629287
hoofdeffect 2	77,4	-338,4610888	493,2610888	-612,3062149	767,1062149
hoofdeffect 3	-952,3	-1355,550975	-549,049025	-1621,092323	-283,5076769
interactie-effect 1-2	-597,9	-1313,675803	117,8758035	-1785,015201	589,2152015
interactie-effect 1-3	136,2	-285,6645287	558,0645287	-563,4629287	835,8629287
interactie-effect 2-3	-77,4	-493,2610888	338,4610888	-767,1062149	612,3062149

Hieruit blijkt dat enkel voor hoofdeffect 3 zowel het 95%- als het 99%-betrouwbaarheidsinterval niet nul bevatten. Daarmee kunnen we zeggen dat het hoofdeffect 3 zowel op het 95%- als het 99%-niveau significant verschillend van nul is. Het is een negatieve waarde wat wil zeggen dat totale kosten lager zijn indien de kans op *local search* 80% bedraagt. De overige effecten zijn niet significant verschillend van nul.

De beste oplossing die bij de parametercombinaties gevonden wordt opent 19 vestigingen. De totale kost bedraagt 380 596.

Groot-geclusterd-identiek

De beste oplossing die overal opduikt omvat een totale kost van 3 732 750 waarbij 39 vestigingen worden geopend.

Groot-willekeurig-identiek

De beste oplossing bedraagt telkens 2848717 en opent 41 vestigingen.

6.3 Conclusies

Het valt op dat bij 10 van de 12 datasets de sturingsparameterwaarden geen invloed hebben op het resultaat. Hiervoor worden enkele mogelijke redenen gegeven. Vervolgens worden de twee datasets waar de resultaten wél varieerden besproken. Tot slot worden de effecten van de overige parameters verduidelijkt.

6.3.1 Geen invloed van sturingsparameterwaarden

Kleine en grote onderneming tegenover middelgrote onderneming

De kleine en grote ondernemingen hebben altijd dezelfde beste oplossing terwijl de middelgrote onderneming dit enkel ondervindt wanneer de klanten in clusters liggen. Een eerste aspect dat hierbij in rekening kan gebracht worden is de verhouding van de totale klantenvraag tot de totale beschikbare capaciteit. Voor de kleine en grote onderneming bedragen deze verhoudingen respectievelijk 64% en 68% wanneer de potentiële vestigingen variërende capaciteiten hebben. Wanneer de potentiële vestigingen identieke capaciteiten hebben zijn deze verhoudingen respectievelijk 62% en 74%. De verhoudingen voor de middelgrote onderneming, namelijk 31% in het eerste geval en 38% in het tweede geval, zijn veel kleiner. Een kleinere verhouding wijst op meer mogelijke toelaatbare oplossingen. Het memetisch algoritme kan meer spelen met het openen en sluiten van de vestigingen en vindt daarom niet dadelijk de beste oplossing.

Middelgrote onderneming met geclusterde klanten tegenover willekeurig verspreide klanten

Wanneer klanten in clusters gelegen zijn, neemt de vraag in die clusters een hoge concentratie aan in tegenstelling tot de gebieden buiten de clusters. Daar liggen geen klanten en wordt bijgevolg niets gevraagd. Vestigingen die in de buurt van de clusters gelegen zijn en over voldoende capaciteit beschikken, hebben daardoor een aanzienlijke kans om geopend te worden. Transportkosten, de variabele kosten, kunnen om die reden laag gehouden worden. Wanneer de klanten verspreid liggen is dit niet zozeer het geval. De vraag ligt meer verspreid over de hele oppervlakte. Alle vestigingen hebben min of meer een redelijke kans om geopend te worden.

Bij de willekeurige datasets kan gemakkelijker rekening gehouden worden met een aantal aspecten. Zo worden vestigingen die beschikken over een voldoende capaciteit met een zo laag mogelijke vaste kost vermoedelijk geopend. Bovendien kan rekening gehouden worden met de ligging van de grootste klanten zodat de transportkosten geminimaliseerd kunnen worden. Hierdoor heeft het memetisch algoritme meer mogelijkheid om goed toelaatbare oplossingen te zoeken. Dit kan het verschil verklaren waarom enkel bij de willekeurige dataset verschillende beste oplossingen te voorschijn komen.

6.3.2 Middelgrote onderneming met willekeurig verspreide klanten

Enkel voor deze ‘categorie’ worden variërende resultaten verkregen. Met zekerheid kunnen hier weinig conclusies over gemaakt worden. Enkel het hoofdeffect van de factor ‘kans op *local search’* is in beide gevallen statistisch significant bij een betrouwbaarheidsinterval van 95%. Bovendien kan bij de identieke dataset met 99% zekerheid gezegd worden dat de kans op *local search* best 80% aanneemt.

Bij de dataset waar de capaciteit van de potentiële vestigingen varieert, is het hoofdeffect van de factor ‘populatiegrootte’ significant verschillend van nul voor een betrouwbaarheidsinterval van 95%. We zijn 95% zeker dat het voordeliger is om een

populatiegrootte van 40 te gebruiken in het memetisch algoritme. Dit kan echter niet gezegd worden met 99% zekerheid. Dan is het effect niet significant verschillend van nul.

6.3.3 Invloeden van de overige parameters

Hier wordt dezelfde structuur aangehouden als in 5.2, experimentele opzet. Vooreerst wordt een overzichtelijke tabel weergegeven. In de eerste kolom staan de 12 datasets. De gegevens die per dataset worden weergegeven zijn de gegevens van de best gevonden oplossing voor elke dataset.

Tabel 6.7: Overzichtstabel resultaten onderzoek

		geopende vestigingen			gesloten vestigingen			
	totale vraag	aantal	capaciteit	vaste kost	aantal	capaciteit	vaste kost	TOTALE KOST
klein-geclusterd-variërend	226	8	241	25916	4	110	11242	161129
klein-geclusterd-identiek	226	8	240	24000	4	120	12000	163203
klein-willekeurig-variërend	226	10	280	31041	2	71	6117	142881
klein-willekeurig-identiek	226	10	300	30000	2	60	6000	134846
middel-geclusterd-variërend	757	14	847	88281	26	1563	160297	456653
middel-geclusterd-identiek	757	16	800	80000	24	1200	120000	490493
middel-willekeurig-variërend	757	16	973	102457	24	1437	146121	400074
middel-willekeurig-identiek	757	19	950	95000	21	1050	105000	380596
groot-geclusterd-variërend	4077	35	4290	444782	15	1641	177701	3647243
groot-geclusterd-identiek	4077	39	4290	429000	11	1210	121000	3732750
groot-willekeurig-variërend	4077	40	4723	490283	10	1208	132200	2888069
groot-willekeurig-identiek	4077	41	4510	451000	9	990	99000	2848717

Ruimtelijke parameters

Als we nagaan voor de identieke datasets hoeveel vestigingen elke keer geopend zijn, kunnen we waarnemen dat voor de geclusterde identieke datasets steeds net voldoende vestigingen geopend zijn om aan de vraag te voldoen. Voor de willekeurige identieke datasets geldt dit niet. Bij de kleine en grote dataset zijn twee vestigingen extra geopend, bij de middelgrote dataset zijn dit drie vestigingen.

Tabel 6.8: Overzicht geclusterd – willekeurig voor de identieke datasets wat de capaciteiten betreft

	totale vraag	totale geopende capaciteit	overbodige capaciteit	capaciteit/ vestiging
klein-geclusterd	226	240	14	30
klein-willekeurig	226	300	74	30
middel-geclusterd	757	800	43	50
middel-willekeurig	757	950	193	50
groot-geclusterd	4077	4290	213	110
groot-willekeurig	4077	4510	433	110

Bij de willekeurige dataset is het voordeliger om over extra capaciteit te beschikken dan net voldoende capaciteit (tabel 6.8). Wanneer alle klanten aan de vestigingen worden toegewezen, kan de totale kost berekend worden. Die blijkt voor de willekeurige datasets steeds lager te liggen dan voor de geclusterde datasets. Het is voor de willekeurige datasets voordeliger om extra vestigingen te openen zodat elke klant een vestiging in de buurt heeft en de transportkosten laag gehouden worden.

Uit tabel 6.7 kan bovendien het volgende afgeleid worden voor de identieke datasets:

Tabel 6.9: Vergelijking willekeurig – geclusterd voor de identieke datasets.

aantal geopende vestigingen	willekeurig > geclusterd
capaciteit geopende vestigingen	willekeurig > geclusterd
vaste kost geopende vestigingen	willekeurig > geclusterd
totale kost	geclusterd > willekeurig

Hieruit kan afgeleid worden dat de transportkosten voor de willekeurige datasets lager liggen dan de transportkosten voor de geclusterde datasets. Het is voor de geclusterde datasets logischer om enkel dichtbijgelegen vestigingen te openen (zie 6.3.1). De grafieken van de geclusterde datasets in bijlage V laten zien dat steeds een aantal vestigingen geopend worden die verder van de clustercentra verwijderd zijn. Deze vestigingen zijn bijgevolg verantwoordelijk voor de hoge transportkosten. Indien voldoende potentiële vestigingen in de nabije buurt van de clusters zouden liggen, zouden deze vestigingen geopend worden. Dit zou tot voordeligere oplossingen leiden.

Wat de 'variërende capaciteiten'-datasets betreft, gelden dezelfde redeneringen en conclusies. Tabel 6.9 geldt ook voor deze datasets. Alleen kan niet onmiddellijk gezegd worden hoeveel 'extra' vestigingen geopend zijn in de willekeurige en geclusterde versies.

Klantgebonden parameters

De vraag van de klant wijzigt niet per ondernemingsgrootte. In dit onderzoek kan geen invloed van de vraag afgeleid worden.

Vestigingsparameters

Deze duiden op de capaciteiten van de potentiële vestigingen. Dit heeft in dit onderzoek soms een invloed op het resultaat. Wat de kleine onderneming betreft, worden in de twee gevallen telkens dezelfde potentiële vestigingen geopend en gesloten. Hier brengt de vestigingsparameter geen verschil teweeg. Dit geldt niet voor de middelgrote en grote ondernemingen. Daar is sprake van kleine verschillen. Deze verschillen blijken willekeurig.

Kostparameters

De totale kost bestaat uit twee componenten. De vaste kost hangt af van de capaciteit van de potentiële vestigingen en wijzigt daarbuiten niet.

De variabele kost wordt berekend door de euclidische afstand tussen elke klant en elke potentiële vestiging te vermenigvuldigen met een kost/km per eenheid en het aantal eenheden dat van de betreffende vestiging naar die klant getransporteerd moeten worden. De kost van 30/km per eenheid wordt tijdens het onderzoek niet veranderd. Allicht worden andere resultaten verkregen indien deze kost van 30 sterk verhoogd of verlaagd zou worden.

7. Algemeen besluit

Het doel van deze eindverhandeling bestond uit het onderzoeken van de parameterwaarden van het memetisch algoritme voor een vestigingsprobleem met capaciteitsbeperking. Dit vestigingsprobleem is een wiskundig moeilijk oplosbaar probleem. Hier werd getracht om met behulp van een memetische algoritme tot een goede oplossing te komen. Nadien werd bestudeerd of de parameters enig effect hadden op de resultaten.

Wat de grote en de kleine onderneming betreft, worden geen verschillende resultaten bekomen. Alle effecten zijn gelijk aan nul. Een wijziging in de parameterwaarden heeft alzo geen invloed op de totale kost. Dit geldt ook voor de datasets van de middelgrote onderneming waar de klanten in clusters liggen.

Zowel bij de dataset waar de potentiële vestigingen variërende capaciteiten hebben als waar de potentiële vestigingen over identieke capaciteiten beschikken, heeft het hoofdeffect van de kans op *local search* een significante invloed op het 95%-niveau. Wanneer de kans 80% is, leidt dit ten opzichte van een kans van 20% tot een vermindering in de totale kosten van beide datasets. Enkel bij de dataset met identieke capaciteiten blijft dit significant op het 99%-niveau. Bij de dataset waar de potentiële vestigingen variërende capaciteiten hebben is het hoofdeffect horende bij een stijging van de populatiegrootte significant op het 95%-niveau. Meer bepaald, bij een stijging van 20 naar 40 individuen, daalt gemiddeld de totale kost. De overige effecten zijn allemaal niet significant verschillend van nul.

Verschillende verklaringen kunnen gegeven worden voor de behaalde resultaten. De verhouding van de totale vraag tot de totale beschikbare capaciteit van een dataset kan van invloed zijn. Bij de kleine en grote datasets bedragen deze verhoudingen 62% tot 74%. Voor de middelgrote datasets zijn deze verhoudingen 31% en 38%. Een lage verhouding geeft het algoritme meer toelaatbare oplossingen wat leidt tot meer rekenwerk. Een tweede verklaring voor de resultaten wordt gezocht bij de ligging van de klanten. Wanneer de klanten in clusters liggen, worden de vestigingen geopend die het dichtst bij de clustercentra gelegen zijn. Indien een verder gelegen vestiging zou geopend worden, lopen de transportkosten ineens hoog op.

Bij de willekeurig verspreide klanten kan de transportkost geminimaliseerd worden door meer vestigingen te openen.

Het is niet uitgesloten dat het memetisch algoritme altijd in een lokaal optimum terechtkomt. Zo kan het bij de geclusterde versies de ‘stap’ niet nemen om meerdere vestigingen te openen en eventueel de transportkosten te verlagen. Doch lijkt dit onrealistisch wanneer we de resultaten aan de hand van de grafieken bekijken (bijlage V).

Indien een nieuw onderzoek hieromtrent verricht zou worden, is het aan te raden om lagere verhoudingen totale klantenvraag – totale capaciteit op te nemen in de datasets. Zo heeft het algoritme meer speelruimte en kunnen eventueel duidelijkere resultaten wat de effecten betreft achterhaald worden. Bij de geclusterde datasets zou het interessant zijn om de potentiële vestigingen eveneens in de clusters te situeren. Die cluster kan hetzelfde middelpunt hebben maar eventueel een grotere straal. De meerderheid van de potentiële vestigingen zou, zoals in de realiteit, in het randgebied van de clusters gesitueerd worden.

Lijst van geraadpleegde werken

Akinc, U. and Khumuwala, B.M. (1977) 'An efficient branch and bound algorithm for the capacitated warehouse location problem', *Management Science*, 23:6, p. 585-594.

Anderson, D.R., Sweeney, D.J. & Williams, T.A. (2000) *Statistiek voor Economie en Bedrijfskunde,* Schoonhoven, Academic Service.

Ballou, H.R. (1973) *Business Logistics Management*, New Jersey, Prentice-Hall Inc.

Chase, R.B, Aquilano, N.J. & Jacobs, F.R. (2004) *Operations Management for Competitive Advantage* (10th edn), New York, McGraw-Hill.

Cobb, G.W. (1998) *Introduction to Design and Analysis of Experiments,* New York, Springer-Verslag.

Cornuejols, G., Nemhauser, G.L. & Wolsey, L.A. (1990) 'The Uncapacitated Facility Location Problem' *in* Mirchandani, P.B. & Francis, R.L. (red.), *Discrete Location Theory*, New York, John Wiley & Sons, Inc, p. 119-172.

Cornuejols, G., Sridharan, R. & Thizy, J.M. (1991) 'A comparison of heuristics and relaxations for the capacitated plant location problem', *European Journal of Operational Research*, 50, p. 280-297.

Coyle, J.J., Bardi, C. & Langley, J. (1992) T*he Management of Business Logistics* (5th edn), St. Paul, West Publishing Company.

Current, J., Daskin, M.S. & Schilling, D. (2002) 'Discrete Network Location Models' *in* Drezner, Z. & Hamacher, H.W. (red.), *Facility Location: Applications and Theory*, Berlijn, Springer-Verslag, p. 81-118.

Daskin, M.S. (1995) *Network and Discrete Location: Models, Algorithms, and Applications*, New York, John Wiley & Sons, Inc..

Digalakis, J. & Margaritis, K. (2004) 'Performance comparison of memetic algorithms', *Applied Mathematics and Computation*, 158:1, p. 237-252.

Eiselt, H.A. & Laporte, G. (1995) 'Objectives in Location Problems' *in* Drezner, Z. & Hamacher, H.W. (red.), *Facility Location: Applications and Theory*, Berlijn, Springer-Verslag, p. 151-180.

Francis, R.L., McGinnis, L.F. and White, J.A. (1983) 'Locational analysis', *European Journal of Operational Research*, 12, p. 220-252.

Hamacher, H.W. & Nickel, S. (1998) 'Classification of location models', *Location Science,* 6, p. 229-242.

Hansen, P. (1987) *Systems of Cities and Facility Location*, Switzerland, Harwood Acadamic Publishers.

Hart, W.E., Krasagnor, N. & Smith, J.E. (2005) 'Memetic Evolutionary Algorithms' *in* Hart, W.E., Krasagnor, N. & Smith, J.E. (red.), *Recent Advances in Memetic Algorithms*, Berlijn, Springer-Verslag, p. 3-27.

Hillier, F.S. & Lieberman, G.J. (2005) *Introduction to Operations Research* (8th edn), New York, McGraw-Hill.

Hromkovič, J. (2004) *Algorithms for Hard Problems: Introduction to Combinatorial Optimization, Randomization, Approximation, and Heuristics* (2nd edn), Berlijn, Springer-Verslag.

Jacobsen, S.K. (1983) 'Heuristics for the capacitated plant location model', *European Journal of Operational Research*, 12, p. 253-261.

Jaramillo, J.H. (1998) 'Genetic algorithms for location problems', Unpublished MS thesis, Department of Industrial Engineering, State University of New York at Buffalo, Buffalo, NY 14260.

Jaramillo, J.H., Bhadury, J. & Batta, R. (2002) 'On the use of genetic algorithms to solve location problems', *Computers & Operations Research*, 29:6, p. 761-779.

Kreher, D.L. & Stinson, D.R. (1999) *Combinatorial Algorithms: Generation, Enumeration, and Search*, Florida, CRC Press LLC.

Kuehn, A.A. & Hamburger, M.J. (1963), 'A heuristic program for locating warehouses', *Management Science*, 9, p. 643-666.

Law, A.M. & Kelton, W.D. (1991) *Simulation Modeling and Analysis* (2nd edn), New York, McGraw-Hill Companies.

Moscato, P. (1999) 'Memetic Algorithms: A Short Introduction' *in* Corne, D., Dorigo, M. & Glover, F. (red.), *New Ideas in Optimization*, Berkshire, McGraw-Hill Publishing Company.

Owen, S.H. & Daskin, M.S. (1998) 'Strategic facility location: A review', *European Journal of Operational Research,* 111:3, p. 423-447.

Pirlot, M. (1996) 'General local search methods', *European Journal of Operational Research*, 29, p. 493-511.

Plastria, F. (1995) 'Continuous Location Problems' *in* Drezner, Z. (red.), *Facility Location: A Survey of Applications and Methods*, New York, Springer-Verslag, p. 225-263.

Reeves, C.R. (1996) 'Modern Heuristic Techniques' *in* Rayward-Smith, V.J., Osman, I.H., Reeves, C.R. & Smith, G.D. (red.), *Modern Heuristic Search Methods,* England-West Sussex, John Wiley & Sons Ltd.

Roodbergen, K.J. (2001) *Layout and Routing Methods for Warehouses,* ERIM Ph. D. series Research in Management 4, TRAIL Research School, Nederland.

Sörensen, K. (2003) 'A framework for robust and flexible optimisation using metaheuristics with applications in supply chain design', doctoraal proefschrift, Faculteit TEW, Universiteit Antwerpen.

Sridharan, R. (1995) 'The capacitated plant location problem', *European Journal of Operational Research*, 87, p. 203-213.

Winston, W.L. (1984) *Operations Research: Applications and Algorithms* (3rd edn), Belmont, CA, Duxbury Press.

Lijst van tabellen

Lijst van figuren

Bijlagen

I Broncode van het memetisch algoritme

II Broncode voor het genereren van de inputdatasets voor het MA

III Lijst van beste oplossingen voor alle sturingsparameter-combinaties per dataset volgens het memetisch algoritme

IV Klantengegevens voor het onderzoek

V Vestigingengegevens en de resultaten

I Broncode van het memetisch algoritme

Common.pas

```
unit Common;

interface

 const
      MaxFacilities   = 100;
      MaxCustomers    = 1001;
      Int64Inf        : Int64 =
9223372036854775807;

 type
      ARRMN        =
array[1..maxfacilities,1..maxcustomers]
of Int64;
      ARRM         =
array[1..maxfacilities] of Int64;
                ARRN       =
array[1..maxcustomers] of Int64;

 VAR

   NrFacilities       : Integer;
   NrCustomers        : Integer;
   FixedCost          : ARRM;
   Capacity           : ARRM;
   Demand             : ARRN;
   Cost,Assignment    : ARRMN;
{Cost[i,j] = Cost to deliver from facility
i to customer j}
   TotalDemand        : Int64;

         popsize                    :
longint = 20; {size of the population}
         ls_prob                    :
real = 0.2;     {local search probability}
   nr_generations             : longint =
10;
   verbosity          : longint = 1; {A parameter
that is used to specify the amount of output.
                              Between 0 and 3,
default is 1.

          verbosity = 0: only output the best
solution, nothing else
                              verbosity = 1: level 0 +
output all intermediate solutions
                              verbosity = 2: level 1 +
output intermediate information about the
memetic algorithm
                              verbosity = 3: level 2 +
output each population}



implementation

end.
```

Io.pas

```
unit io;

interface

 VAR
 OutFile          : Text;
 OutFileName      : String;
 InFileName             : String;

 FUNCTION ReadParams: Boolean;

implementation

 uses Common, sysutils;

PROCEDURE ReadData (InFileStr : String);
  {This procedure reads data in the format of
the OR library (Beasley)}
```

```
 VAR
  F        : Text;
  i, j     : Integer;
  dummy    : Real;
 BEGIN
  Assign(F, InfileStr);
  Reset(F);
  TotalDemand := 0;
  Read(F, NrFacilities);
  ReadLn(F, NrCustomers);
  For i := 1 to NrFacilities do
   BEGIN
   Read(F, dummy);

{FOR NOW!!!! WE NEED TO READ
NUMBERS AFTER THE COMMA!!!
ALL THE TRUNC (DUMMY)'s have
to GO}

    Capacity[i] := trunc(dummy);
    ReadLn(F, dummy);
    FixedCost[i] := trunc(dummy);
    END;
  For i := 1 to NrCustomers do
    BEGIN
    Read(F, dummy);
          Demand[i] := trunc(dummy);
    TotalDemand := TotalDemand +
Demand[i];
    For j := 1 to NrFacilities do
      begin
      Read(F, dummy);
      Cost[j,i] := trunc(dummy);

      {HERE THINGS CAN GO
SERIOUSLY WRONG!!!!}

      {Cost[j,i] := trunc(Cost[j,i] /
Demand[i]);}

                {TRYING IT
DIFFERENTLY: THE COST IS NOW
THE COST TO DELIVER ONE
UNIT}

      end;
    ReadLn(F);
    END;
  {Add a dummy customer (NrCustomers +
1) to the cost matrix
   The cost of servicing this customer is zero
from any facility}
   For j := 1 to NrFacilities do
    Cost[j,NrCustomers+1] := 0;
END; {ReadData}


FUNCTION ReadParams: Boolean;
 VAR
  param   : String;
  i       : Integer;
 BEGIN
  ReadParams := True;

  If ParamStr(1) = '' then
    begin
    writeln ('USAGE: memfacloc infile
outfile [parameters]');
    writeln ('Parameters: (i is an integer, r is a
real number)');
    writeln ('  -p i        : population size');
    writeln ('  -g i        : number of
generations');
    writeln ('  -l r        : local search rate
(between 0 and 1)');
    writeln ('  -v i        : verbosity level
(between 0 and 3)');
    ReadParams := False;
    end
  Else
    begin
    param := 'dummy';
    i := 3;
    InFileName := ParamStr(1);
    ReadData(InFileName);
    OutFileName := ParamStr(2);
    Assign (Outfile, OutFileName);
    Rewrite (Outfile);
    param := ParamStr(i);
    While param <> '' do
      begin
      If param = '-p' then
```

```
        begin
          popsize := strtoint
(paramstr(i+1));
          i := i + 2;
        end
      else
        if param = '-g' then
          begin
            nr_generations := strtoint
(paramstr(i+1));
            i := i + 2
          end
        else
          if param = '-l' then
            begin
              ls_prob := strtofloat
(paramstr(i+1));
              i := i + 2
            end
          else
            if param = '-v' then
              begin
                verbosity :=
strtoint(paramstr(i+1));
                i := i + 2;
              end
            else
              begin
              WriteLn ('Invalid use of
parameters');
              ReadParams := False;
              end;
      param := ParamStr(i);
      end;
    end;
END;

end.
```

Loc.pas

```
UNIT Loc;

{This unit contains
  - the type Location that is a solution to
a location problem
  - a procedure that reads all data in the form
of the OR Library (Beasley)
  - a procedure that initializes a solution
  - a function that evaluates a solution}

{**********************************
**********************}
INTERFACE
{**********************************
**********************}

  uses common;

  TYPE

    Location = Array[1..MaxFacilities] of
Boolean;

  PROCEDURE Evaluate (L : Location; VAR
Evaluation : Int64);

  PROCEDURE InitializeLocation (VAR L :
Location);

  PROCEDURE AddFacility (VAR L :
Location; F : Integer);

  PROCEDURE DropFacility (VAR L :
Location; F : Integer);

  PROCEDURE WriteLocation (VAR F :
Text; L : Location);

  PROCEDURE Random_Location (VAR L :
Location; VAR E : Int64);

{**********************************
**********************}
IMPLEMENTATION
{**********************************
**********************}

USES
  trans;
```

```
PROCEDURE InitializeLocation (VAR
L : Location);
{Sets all facilities of a specific solution
to "closed" (false)}
  VAR
    i     : Integer;
  BEGIN
    For i := 1 to NrFacilities do
      L[i] := False;
  END; {InitializeLocation}

{This is an ordinary evaluation
function}

PROCEDURE Evaluate (L : Location;
VAR Evaluation : Int64);

  VAR
    i,j          :       Integer;

TempEval,TotalCapacity,AllocationCos
t : Int64 ;
    tempCap   :       ARRM;
    tempCost  : ARRMN;
    nrOpenFac : Integer;

  BEGIN
    TempEval := 0;
    TotalCapacity := 0;

    {Set the capacity of the closed
facilities to zero and calculate the total
fixed cost}

    nrOpenFac := 0;
    For i := 1 to NrFacilities do
      If L[i] = true
      then
        Begin
        inc(nrOpenFac);
        tempCap[nrOpenFac] :=
Capacity[i];
        for j := 1 to nrCustomers + 1 do
tempCost[nrOpenFac,j] := Cost[i,j];
        TotalCapacity := TotalCapacity +
tempCap[nrOpenFac];
        TempEval := TempEval + FixedCost[i];
        end;

{    Writeln ('Total location cost : ',
TempEval);
    WriteLn ('Total demand : ', TotalDemand);
    WriteLn ('Total Capacity : ',
TotalCapacity);}

    If TotalCapacity >= TotalDemand then
      BEGIN
      {Demand of the dummy customer:
Demand = TotalCapacity - TotalDemand}
        Demand[NrCustomers + 1] :=
TotalCapacity - TotalDemand;
        {For i := 1 to NrCustomers + 1 do
WriteLn (Demand[i]:10:2,' ');}

Transport(NrOpenFac,NrCustomers+1,tempC
ap,Demand,tempCost,Assignment,Allocation
Cost);
        {WriteLn ('Allocation Cost : ',
AllocationCost:10:2);}
        TempEval := TempEval +
AllocationCost;

      {WRITE THE SOLUTION}
      {nrOpenFac := 0;
      For i := 1 to NrFacilities do
        begin
        if L[i] then
          begin
            inc (nrOpenFac);
            Write (i, ' ');
            For j := 1 to NrCustomers + 1 do
                begin
                Write (Assignment[nrOpenFac,j], '
');
                end;
                Writeln ('Capacity : ',
tempcap[nrOpenFac]);
           end;
        end;
      For j := 1 to NrCustomers + 1 do
        Write (Demand[j], ' ');
      Writeln;
```

```
      Writeln ('TOTAL COST : ',
TempEval);}
      {END WRITE THE SOLUTION}

      END
    Else {TotalCapacity < TotalDemand}
      BEGIN
      TempEval := Int64Inf -
TotalCapacity;
      {WriteLN ('INFEASIBLE
SOLUTION : ',TempEval);}
      END;

    Evaluation := TempEval;

  END;{Evaluate}

  PROCEDURE AddFacility (VAR L :
Location; F : Integer);
    BEGIN
      If L[F] = False
      Then L[F] := True
      Else WriteLn ('ERROR : Facility ',
F, ' is already in this location');
    END;

  PROCEDURE DropFacility (VAR L :
Location; F : Integer);
    BEGIN
      If L[F] = True
      Then L[F] := False
      Else WriteLn ('ERROR : Facility ',
F,' cannot be dropped from this
location');
    END;

  PROCEDURE WriteLocation (VAR F
: Text; L : Location);
    VAR
      i : Integer;
    BEGIN
      For i := 1 to NrFacilities do
        If L[i] = True
        Then Write (F, i, ' ');
    END;

  PROCEDURE Random_Location (VAR L :
Location; VAR E : Int64);
    VAR
      i : Integer;
    BEGIN
      For i := 1 to nrFacilities do
        If random < 0.5 then L[i] := true else L[i]
:= false;
      Evaluate (L, E);
    END;

END. {Main Program}
```

Memfacloc.pas

```
Program memfacloc;
{Memetic algorithm for the capacitated
facility location problem}

uses
  loc,
  common,
  io;

Function BestToAddOrDrop (L : Location;
                    Var Eval: Int64 ) : Integer;

  Var
    i          : Integer;
    BestYet    : Integer;
    BestYetEval : Int64;
    TempL      : Location;
    TempEval   : Int64;
  Begin
    BestYetEval := Int64Inf;
    BestYet := 0;
    For i := 1 To NrFacilities Do
       Begin
         TempL := L;
         If L[i]
           Then
           Begin
             DropFacility (TempL, i);
             Evaluate (TempL, TempEval);
             If TempEval < BestYetEval
               Then
```

```
            Begin
              BestYetEval := TempEval;
              BestYet := i;
            End;
          End
        Else
          Begin
            AddFacility (TempL, i);
            Evaluate (TempL, TempEval);
            If TempEval < BestYetEval
              Then
              Begin
                BestYetEval := TempEval;
                BestYet := i;
              End;
            End
        End;
    If BestYet = 0 Then
      Begin
        WriteLn ('No feasible moves.');
        BestToAddOrDrop :=
random(nrFacilities) + 1;
      End
    Else
      Begin
        Eval := BestYetEval;
        BestToAddOrDrop := BestYet;
      End;
  End;

Procedure AddFacilitiesUntilFeasible
(Var L : Location);

  Var
    totalcap : Int64;
    i : integer;
  Begin
    totalCap := 0;
    i := 1;
    While totalCap < totalDemand Do
      Begin
        AddFacility (L, i);
        writeln ('Adding facility ', i);
        totalCap := totalCap + Capacity[i];
        inc (i);
      End;
  End;

Procedure SteepestDescent (Var L : Location;
Var E : Int64);

  Var
    F             : Integer;
    BestYet       : Location;
    BestYetEval   : Int64;
    BetterFound   : Boolean;
  Begin
    BestYetEval := Int64Inf;
    Repeat
      BetterFound := False;
      F := BestToAddOrDrop(L, E);
      If F <> 0 Then
        Begin
          If L[F] Then DropFacility (L, F)
          Else AddFacility (L,F);
          {WriteLocation (Outfile, L);
          WriteLn (Outfile, ': ', E);}
          If E < BestYetEval
            Then
            Begin
              BestYet := L;
              BestYetEval := E;
              BetterFound := True;
            End;
        End;
      Until BetterFound=FALSE;
      L := BestYet;
      E := BestYetEval;
  End;


Var
  pop_sol : array Of location;
  pop_eval : array Of Int64;
  i, sol1, sol2, p1, p2, r1, r2, x_point,
generation : longint;
  best_sol : location;
  best_eval : int64;

Begin
  If ReadParams Then
    Begin
```

```
    Randomize;
    writeln (
    'Parameters OK. Data (seems to be) succesfully read. Generating initial population.'
    );
    writeln (outfile, 'Memetic algorithm for the capacitated facility location problem.'
    );
    writeln (outfile, 'Data file        : ', infilename);
    writeln (outfile, 'Local search rate : ', ls_prob:6:4);
    writeln (outfile, 'Population size   : ', popsize);
    writeln (outfile, 'Generations       : ', nr_generations);

          {Initialize a few parameters}
    best_eval := int64inf;
    setlength (pop_sol, popsize);
    setlength (pop_eval, popsize);

    If verbosity >= 2 Then writeln (outfile, 'Generating initial population.');

          {Start generating the population}
    For i := 1 To popsize Do
      Begin
        {Generate a random solution at location i in the population and evaluate it}
        random_location(pop_sol[i], pop_eval[i]);
        {Improve it using steepest descent with probability ls_prob}
        If random < ls_prob Then
          Begin
            If verbosity >= 2 Then writeln (outfile,'improving solution ', i);
            steepestdescent(pop_sol[i], pop_eval[i]);
          End;
                {Make a note if this is the best solution ever encountered}
        If pop_eval[i] < best_eval Then
          Begin
            best_sol := pop_sol[i];
            best_eval := pop_eval[i];
          End;
      End;
    {The initial population has been generated randomly}

    If verbosity >= 1 Then
      Begin
        writeln (outfile, 'Initial population');
        For i := 1 To popsize Do
          Begin
            WriteLocation (outfile, pop_sol[i]);
            writeln (outfile, pop_eval[i]);
          End;
      End;

    If verbosity >= 3 Then
      Begin
        writeln (outfile, 'Best solution in the initial population: ');
        WriteLocation (outfile, best_sol);
        writeln (outfile, best_eval);
      End;

    If verbosity >= 2 Then writeln ('Starting memetic algorithm');

          {Start the memetic algorithm for nr_generations generations}
    For generation := 1 To nr_generations Do

      Begin
        writeln ('Generation ', generation);
        If verbosity >= 1 Then
          writeln (outfile, 'GENERATION ', generation);
        If verbosity >= 2 Then
          writeln (outfile,
                'Finding two good solutions using binary tournament selection');
```

```
                {Binary tournament
selection: find two solutions randomly
and take the best one to be p1 (parent
1)}
        sol1 := trunc(random * popsize) +
1;
        sol2 := trunc(random * popsize) +
1;
        If pop_eval[sol1] <
pop_eval[sol2] Then p1 := sol1
        Else p1 := sol2;

                {Binary tournament
selection part 2: find parent 2, but make
sure that none of the candidates for p2
are equal to p1}
        Repeat
          sol1 := trunc(random * popsize)
+ 1
        Until sol1 <> p1;
        Repeat
          sol2 := trunc(random * popsize)
+ 1
        Until sol2 <> p1;
        If pop_eval[sol1] <
pop_eval[sol2] Then p2 := sol1
        Else p2 := sol2;

                {Write some stuff}
        If verbosity >= 2 Then
          Begin
            write (outfile,'Parent 1: ', p1, '
');
            writelocation(outfile,
pop_sol[p1]);
            writeln (outfile,pop_eval[p1]);
            write (outfile, 'Parent 2: ', p2, '
');
            writelocation(outfile,
pop_sol[p2]);
            writeln (outfile,pop_eval[p2]);
          End;

        If verbosity >= 2 Then
          writeln (outfile,'Finding a place
in the population to put new solutions');

                {Negative binary tournament
selection to find solutions to remove, take two
candidates and remove the worst. Note that no
intermediate solutions are used, we insert the
new solutions directly in the population in the
place of the old ones. This might give
problems if the old ones and the new ones are
equal, but I'm not sure of this. Worth it to find
it out...}
        sol1 := trunc(random * popsize) + 1;
        sol2 := trunc(random * popsize) + 1;
        If pop_eval[sol1] > pop_eval[sol2]
Then r1 := sol1
        Else r1 := sol2;

                {Negative bts part II, make
sure that the candidate solutions to remove are
not equal to the first solution to remove}
        Repeat
          sol1 := trunc(random * popsize) + 1
        Until sol1 <> r1;
        Repeat
          sol2 := trunc(random * popsize) + 1
        Until sol2 <> r1;
        If pop_eval[sol1] > pop_eval[sol2]
Then r2 := sol1
        Else r2 := sol2;

                {Write some stuff}
        If verbosity >= 2 Then
          Begin
            write (outfile,'Remove 1: ', r1, '   ');
            writelocation(outfile, pop_sol[r1]);
            writeln (outfile,pop_eval[r1]);
            write (outfile,'Remove 2: ', r2, '   ');
            writelocation(outfile, pop_sol[r2]);
            writeln (outfile,pop_eval[r2]);

            writeln (outfile,
                  'Crossing over the two parent
solutions using one-point crossover')
                ;
          End;
```

```
                {Generate the crossover
point at random}
        x_point := trunc(random *
(nrfacilities - 1)) + 2;

        If verbosity >= 2 Then
          writeln (outfile,'Crossover point
: ', x_point);

                {Crossover, copy the
first part of p1 to r1 and the second to
r2, etc.}
        For i := 1 To x_point - 1 Do
          Begin
            pop_sol[r1][i] :=
pop_sol[p1][i];
            pop_sol[r2][i] :=
pop_sol[p2][i];
          End;
        For i := x_point To nrfacilities Do
          Begin
            pop_sol[r1][i] :=
pop_sol[p2][i];
            pop_sol[r2][i] :=
pop_sol[p1][i];
          End;

        If verbosity >= 2 Then
          writeln (outfile,'Evaluating the
new solutions.');

                {Evaluate both
solutions}
        evaluate (pop_sol[r1],
pop_eval[r1]);
        evaluate (pop_sol[r2],
pop_eval[r2]);
        If verbosity >= 2 Then
          writeln (outfile,'The new
solutions:');
        If verbosity >= 2 Then
          Begin
            writelocation (outfile,
pop_sol[r1]);
            writeln (outfile, pop_eval[r1]);
            writelocation (outfile, pop_sol[r2]);
            writeln (outfile, pop_eval[r2]);
          End;

        {Improve the first offspring with
probability ls_prob}
        If random < ls_prob Then
          Begin
            If verbosity >= 2 Then writeln
(outfile, 'Improving new solution 1.');
            steepestdescent (pop_sol[r1],
pop_eval[r1]);
          End;

        If verbosity >= 2 Then write (outfile,
                'Checking to see if solution
1 is the best one encountered. '
          );

                {See if offspring 1 is the best
solution encountered}
        If pop_eval[r1] < best_eval Then
          Begin
            If verbosity >= 2 Then writeln
(outfile,'It is!');
            best_sol := pop_sol[r1];
            best_eval := pop_eval[r1];
          End
        Else If verbosity >= 2 Then writeln
(outfile,'It is not.');

        {Improve offspring 2 with prob.
ls_prob}
        If random < ls_prob Then
          Begin
            If verbosity >= 2 Then writeln
(outfile, 'Improving new solution 2.');
            steepestdescent (pop_sol[r2],
pop_eval[r2]);
          End;

        If verbosity >= 2 Then write (outfile,
                'Checking to see if solution
2 is the best one encountered. '
          );
```

```
                    {See if it is the best
solution yet encountered}
        If pop_eval[r2] < best_eval Then
          Begin
           If verbosity >= 2 Then writeln
(outfile,'It is!');
            best_sol := pop_sol[r2];
            best_eval := pop_eval[r2];
          End
        Else If verbosity >= 2 Then
writeln (outfile,'It is not.');

        {Write some stuff}
        If verbosity >= 1 Then
          Begin
           writeln (outfile, 'The following
solutions are added to the population.');
            writelocation (outfile,
pop_sol[r1]);
            writeln (outfile, pop_eval[r1]);
            writelocation (outfile,
pop_sol[r2]);
            writeln (outfile, pop_eval[r2]);
          End;

                    {If the verbosity level is
really high, write the new solution.}
        If verbosity >= 3 Then
          Begin
           writeln (outfile,'Population
after generation ', generation);
            For i := 1 To popsize Do
              Begin
                WriteLocation (outfile,
pop_sol[i]);
               writeln (outfile,
pop_eval[i]);
              End;
          End;

      End; {End of the memetic
algorithm main loop}

    WriteLn (OutFile, 'BEST
SOLUTION FOUND');
    writelocation (outfile, best_sol);
    writeln (outfile, best_eval);
    Close(Outfile);
  End;

  writeln ('All done. Output written on ',
outfilename);
End.
```

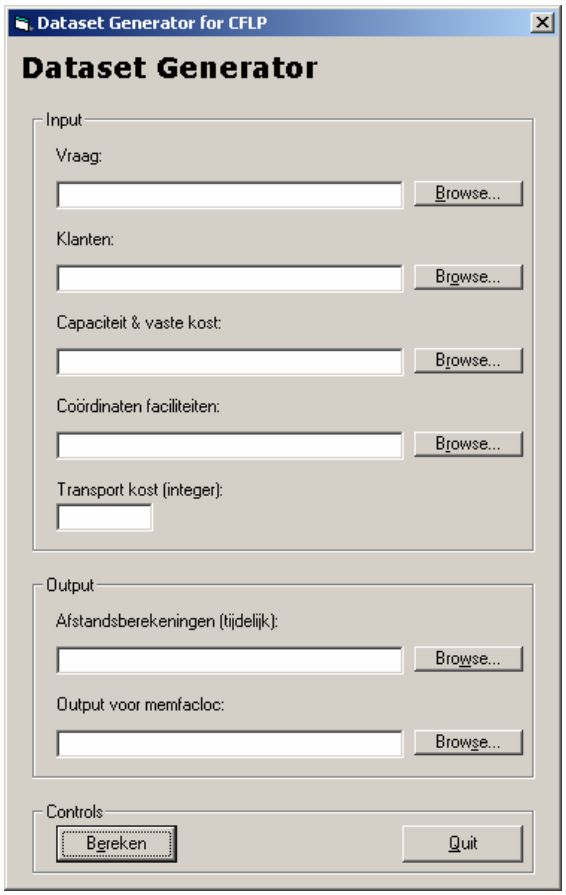
Trans.pas

```
unit Trans;

interface

 uses common;

PROCEDURE TRANSPORT(
      M,N    :INTEGER;
                A      :ARRM;
                B      :ARRN;
                C          :ARRMN;
  VAR  X      :ARRMN;
  VAR  KO     :INT64);

implementation

PROCEDURE TRANSPORT(
      M,N    :INTEGER;
                A      :ARRM;
                B      :ARRN;
                C      :ARRMN;
  VAR  X      :ARRMN;
  VAR  KO     :INT64);

  VAR I,J: INTEGER;
          SF,R,RA  : INT64;
     LAB,LAB1,LAB2:BOOLEAN;
     U,W,EPS      :ARRM;
     V,K,DEL      :ARRN;
BEGIN                { INITIALIZATION OF
DUAL VARIABLES U AND V }
  FOR I:=1 TO M DO U[I]:=0;
  FOR J:=1 TO N DO BEGIN
    R:=Int64Inf;
    FOR I:=1 TO M DO BEGIN
      X[I,J]:=0;  SF:=C[I,J];
```

```
      IF SF < R THEN R:=SF
    END;
    V[J]:=R
  END;  { FOR J }
  LAB1:=FALSE;
  REPEAT  { UNTIL LAB1 }    {
LAB1 IS TRUE IF THE OPTIMAL
SOLUTION }
                    { HAS BEEN
FOUND AND FALSE OTHERWISE }
    FOR I:=1 TO M DO BEGIN
      W[I]:=0;  EPS[I]:=A[I]
    END;
    FOR J:=1 TO N DO BEGIN
      K[J]:=0;  DEL[J]:=0
    END;
    REPEAT  { UNTIL LAB }
      LAB:=TRUE;  LAB2:=TRUE;
{ LABELING ROWS AND
COLUMNS }
      I:=0;
      REPEAT  { UNTIL (I = M) OR
(LAB2 = FALSE) }
        I:=I+1;
        SF:=EPS[I];  EPS[I]:=-SF;
        IF SF > 0 THEN BEGIN
{ ROW I BECOMES LABELED }
          RA:=U[I];  J:=0;
          REPEAT  { UNTIL (J = N)
OR (LAB2 = FALSE) }
            J:=J+1;
            IF (DEL[J] = 0) AND (V[J]-
RA = C[I,J]) THEN BEGIN
              K[J]:=I;          {
COLUMN J CAN BE LABELED }
              DEL[J]:=SF;
LAB:=FALSE;
              IF B[J] > 0 THEN BEGIN
{ BREAKTHROUGH }
                LAB:=TRUE;
LAB2:=FALSE;
                SF:=ABS(DEL[J]);
                R:=B[J];
                IF R < SF THEN SF:=R;
                B[J]:=R-SF;
                REPEAT
                  I:=K[J];  X[I,J]:=X[I,J]+SF;
                  J:=W[I];
                  IF J <> 0 THEN
X[I,J]:=X[I,J]-SF
                UNTIL J = 0;
                A[I]:=A[I]-SF;  J:=0;
                REPEAT
                  J:=J+1;  LAB1:=B[J] <= 0
                UNTIL (J = N) OR NOT
LAB1;
                IF LAB1 THEN BEGIN
                      { OPTIMAL
SOLUTION HAS BEEN FOUND }
                  SF:=0;
                  FOR I:=1 TO M DO
                    FOR J:=1 TO N DO
BEGIN
                      R:=X[I,J];
                      IF R > 0 THEN
SF:=SF+R*C[I,J]
                    END ;
                  KO:=SF
                END { IF LAB1 }
              END  { IF B[J] > 0 }
            END  { IF (DEL[J] = 0) ... }
          UNTIL (J = N) OR NOT LAB2
        END  { IF SF > 0 }
      UNTIL (I = M) OR NOT LAB2;
      IF NOT LAB THEN BEGIN        {
LABELING ROWS FROM COLUMNS }
        LAB:=TRUE;
        FOR J:=1 TO N DO BEGIN
          SF:=DEL[J];
          IF SF > 0 THEN BEGIN
            FOR I:=1 TO M DO
              IF EPS[I] = 0 THEN BEGIN
                R:=X[I,J];
                IF R > 0 THEN BEGIN
                  W[I]:=J;
                  IF R <= SF THEN
EPS[I]:=R
                  ELSE EPS[I]:=SF;
                  LAB:=FALSE
                END  { IF R > 0 }
              END;  { IF EPS[I] = 0, FOR I }
            DEL[J]:=-SF
```

```
          END { IF SF > 0 }
        END  { FOR J }
      END  { IF NOT LAB }
    UNTIL LAB;                          {
END OF LABELING }
    IF LAB2 THEN BEGIN                  {
MODIFYING DUAL VARIABLES }
      R:=Int64Inf;
      FOR I:=1 TO M DO
        IF EPS[I] <> 0 THEN BEGIN
          RA:=U[I];
          FOR J:=1 TO N DO
            IF DEL[J] = 0 THEN
BEGIN
                SF:=C[I,J]+RA-V[J];
                IF R > SF THEN R:=SF
              END
          END;  { IF EPS[I] <> 0, FOR I }
        FOR I:=1 TO M DO IF EPS[I] = 0
    THEN U[I]:=U[I]+R;
        FOR J:=1 TO N DO IF DEL[J] = 0
    THEN V[J]:=V[J]+R
      END  { IF LAB2 }
    UNTIL LAB1
END;  { TRANSPORT }


end.
```

II Broncode voor het genereren van de inputdatasets voor het MA




```
VERSION 5.00
Object = "{F9043C88-F6F2-101A-A3C9-
08002B2F49FB}#1.2#0"; "comdlg32.ocx"
Begin VB.Form frmMain
   BorderStyle     =   1  'Fixed Single
   Caption         =   "Dataset Generator for
CFLP"
   ClientHeight    =   8535
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   5520
   LinkTopic       =   "Form1"
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   8535
   ScaleWidth      =   5520
   StartUpPosition =   3  'Windows Default
   Begin MSComDlg.CommonDialog cmdlg
      Left           =   4800
      Top            =   120
      _ExtentX       =   847
      _ExtentY       =   847
      _Version       =   393216
   End
   Begin VB.Frame Frame3
      Caption         =   "Controls"
      Height          =   735
      Left            =   240
      TabIndex        =   3
      Top             =   7680
      Width           =   5055
      Begin VB.CommandButton cmdBereken
         Caption         =   "B&ereken"
         Height          =   375
         Left            =   240
         TabIndex        =   16
         Top             =   240
         Width           =   1215
      End
      Begin VB.CommandButton cmdQuit
         Caption         =   "&Quit"
         Height          =   375
         Left            =   3720
         TabIndex        =   4
         Top             =   240
         Width           =   1215
      End
   End
   Begin VB.Frame Frame2
      Caption         =   "Output"
      Height          =   2055
      Left            =   240
      TabIndex        =   2
      Top             =   5400
      Width           =   5055
      Begin VB.TextBox txtOutput
         Height          =   285
         Index           =   1
         Left            =   240
         TabIndex        =   18
         Top             =   1560
         Width           =   3495
      End
      Begin VB.CommandButton cmdBrowseOut
         Caption         =   "Brow&se..."
         Height          =   255
         Index           =   1
         Left            =   3840
         TabIndex        =   17
         Top             =   1560
         Width           =   1095
      End
```

```
      Begin VB.CommandButton
cmdBrowseOut
         Caption         =   "Bro&wse..."
         Height          =   255
         Index           =   0
         Left            =   3840
         TabIndex        =   15
         Top             =   720
         Width           =   1095
      End
      Begin VB.TextBox txtOutput
         Height          =   285
         Index           =   0
         Left            =   240
         TabIndex        =   14
         Top             =   720
         Width           =   3495
      End
      Begin VB.Label lblInput
         Caption         =   "Output voor
memfacloc:"
         Height          =   255
         Index           =   4
         Left            =   240
         TabIndex        =   20
         Top             =   1200
         Width           =   2295
      End
      Begin VB.Label lblInput
         Caption         =
"Afstandsberekeningen (tijdelijk):"
         Height          =   255
         Index           =   3
         Left            =   240
         TabIndex        =   19
         Top             =   360
         Width           =   2295
      End
   End
   Begin VB.Frame Frame1
      Caption         =   "Input"
      Height          =   4455
      Left            =   240
      TabIndex        =   1
      Top             =   720
      Width           =   5055
      Begin VB.TextBox txtTransport
         Height          =   285
         Left            =   240
         TabIndex        =   25
         Top             =   3960
         Width           =   975
      End
      Begin VB.TextBox txtInput
         Height          =   285
         Index           =   3
         Left            =   240
         TabIndex        =   22
         Top             =   3240
         Width           =   3495
      End
      Begin VB.CommandButton cmdBrowse
         Caption         =   "B&rowse..."
         Height          =   255
         Index           =   3
         Left            =   3840
         TabIndex        =   21
         Top             =   3240
         Width           =   1095
      End
      Begin VB.CommandButton cmdBrowse
         Caption         =   "Br&owse..."
         Height          =   255
         Index           =   2
         Left            =   3840
         TabIndex        =   12
         Top             =   1560
         Width           =   1095
      End
      Begin VB.TextBox txtInput
         Height          =   285
         Index           =   2
         Left            =   240
         TabIndex        =   11
         Top             =   1560
         Width           =   3495
      End
      Begin VB.CommandButton cmdBrowse
         Caption         =   "B&rowse..."
         Height          =   255
         Index           =   1
         Left            =   3840
         TabIndex        =   9
         Top             =   2400
         Width           =   1095
      End
      Begin VB.TextBox txtInput
         Height          =   285
         Index           =   1
         Left            =   240
         TabIndex        =   8
         Top             =   2400
         Width           =   3495
      End
      Begin VB.CommandButton cmdBrowse
         Caption         =   "&Browse..."
         Height          =   255
         Index           =   0
         Left            =   3840
         TabIndex        =   6
```

```
      Top           =  720
      Width         =  1095
    End
    Begin VB.TextBox txtInput
      Height        =  285
      Index         =  0
      Left          =  240
      TabIndex      =  5
      Top           =  720
      Width         =  3495
    End
    Begin VB.Label lblInput
      Caption       =  "Transport kost
(integer):"
      Height        =  255
      Index         =  6
      Left          =  240
      TabIndex      =  24
      Top           =  3720
      Width         =  2295
    End
    Begin VB.Label lblInput
      Caption       =  "Coördinaten
faciliteiten:"
      Height        =  255
      Index         =  5
      Left          =  240
      TabIndex      =  23
      Top           =  2880
      Width         =  2295
    End
    Begin VB.Label lblInput
      Caption       =  "Klanten:"
      Height        =  255
      Index         =  2
      Left          =  240
      TabIndex      =  13
      Top           =  1200
      Width         =  2295
    End
    Begin VB.Label lblInput
      Caption       =  "Capaciteit && vaste
kost:"
      Height        =  255
      Index         =  1
      Left          =  240
      TabIndex      =  10
      Top           =  2040
      Width         =  2295
    End
    Begin VB.Label lblInput
      Caption       =  "Vraag:"
      Height        =  255
      Index         =  0
      Left          =  240
      TabIndex      =  7
      Top           =  360
      Width         =  2295
    End
  End
  Begin VB.Label Label1
    Caption         =  "Dataset Generator"
    BeginProperty Font
      Name          =  "Verdana"
      Size          =  14.25
      Charset       =  0
      Weight        =  700
      Underline     =  0   'False
      Italic        =  0   'False
      Strikethrough =  0   'False
    EndProperty
    Height          =  495
    Left            =  120
    TabIndex        =  0
    Top             =  120
    Width           =  3135
  End
End
Attribute VB_Name = "frmMain"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Sub cmdBrowse_Click(Index As Integer)
   cmdlg.Filter = "Text (*.txt) | *.txt"
   cmdlg.ShowOpen
   Me.txtInput(Index).Text = cmdlg.FileName
End Sub

Private Sub cmdBrowseOut_Click(Index As
Integer)
   cmdlg.Filter = "Text (*.txt) | *.txt"
   cmdlg.ShowSave
   Me.txtOutput(Index).Text = cmdlg.FileName
End Sub

Private Sub cmdQuit_Click()
   End
End Sub

Private Sub cmdBereken_Click()
On Error GoTo ErrRoutine

   Dim strFO, strFO2, strFO3, strBuff As String
   Dim lijnTeller, teller0, teller1, teller2 As Integer

   ''''''''''''''''''''''''''''''''
   ' Bestand openen, lijnen tellen '
   ''''''''''''''''''''''''''''''''
```

```
    For x = 0 To 2
        strFO = Me.txtInput(x).Text

        Open strFO For Input As #1
            lijnTeller = 0
            Do Until EOF(1)
                Line Input #1, strBuff
                lijnTeller = lijnTeller + 1
            Loop
            lijnTeller = lijnTeller - 1     ' nu
hebben we 't juiste aantal lijnen (zonder de
header)
        Close #1

        Select Case x
        Case 0:
            teller0 = lijnTeller
        Case 1:
            teller1 = lijnTeller
        Case 2:
            teller2 = lijnTeller
        End Select
    Next x

    ' lijnen checken '
    If teller0 <> teller2 Then
        MsgBox "Er is wat mis met de
inputfiles, check of ze evenveel klanten als
vraag bevatten.", vbCritical, "Fout
gevonden"
        Exit Sub
    End If

    Dim iFac, iKlant As Integer
    iFac = teller1
    iKlant = teller0

    """""""""""""""""""""""
    ' bestand openen voor mfl '
    ' lijnen opslaan          '
    ' Cap en kost opslaan     '
    """""""""""""""""""""""
    Dim strbuff2 As Variant
    strFO = Me.txtOutput(1).Text    ' output
voor memfacloc
    strFO2 = Me.txtInput(1).Text    '
Capaciteit & vaste kost
        Open strFO For Output As #1
        Open strFO2 For Input As #2
            Print #1, CStr(iFac) & " " &
CStr(iKlant)
            Line Input #2, strBuff  ' eerste lijn,
met header, die we toch niet nodig hebben
            Do Until EOF(2)         ' Loopke, dat
alle lijn uit cap-kost plakt in het outputfile
                Line Input #2, strBuff
                strbuff2 = Split(strBuff, vbTab)
                Print #1, strbuff2(0) & " " & strbuff2(1)   '
plak de gelezen lijn in 't outputbestand
            Loop                ' einde loop
        Close #2                ' sluit bestandje 2
        Close #1                ' sluit bestandje 1


    """""""""""""""""""""""
    ' Berekening eucl afstd  '
    ' tussen per klant en    '
    ' elke facility in temp  '
    """""""""""""""""""""""
    'klant = 2, fac = 3

    Dim xK, yK, xF, yF As Integer
    Dim eucl As Double
    Dim afstdLijn As String
    afstdLijn = ""

    strFO = Me.txtOutput(0).Text ' temp file met
afstanden
    strFO2 = Me.txtInput(2).Text ' coord. klanten
    strFO3 = Me.txtInput(3).Text ' coord. fac


        Open strFO For Output As #1
        Open strFO2 For Input As #2
            Line Input #2, strBuff  ' eerste regel met
header, hebben we toch niet nodig

            Do Until EOF(2)         ' heel 't klantencoord
doorlopen
                Line Input #2, strBuff
                strbuff2 = Split(strBuff, vbTab)
                xK = CInt(strbuff2(0))      ' x coord van
klant
                yK = CInt(strbuff2(1))      ' y coord van
klant

                    Open strFO3 For Input As #3
'bestandje met fac-coords openenen
                    Line Input #3, strBuff  ' eerste lijn met
headers, mogen we wegsmijten
                    Do Until EOF(3)         ' de hele fac-
coords doorlopen, telkens afstand berekenen
                        Line Input #3, strBuff
                        strbuff2 = Split(strBuff, vbTab)
                        xF = CInt(strbuff2(0))  ' x coord van
fac
                        yF = CInt(strbuff2(1))  ' y coord van
fac
                            ' EUCL afstand berekenen,
OUTPUTTEN naar temp-file
```

```
                    eucl = Round(Sqr((xF -
xK) ^ 2 + (yF - yK) ^ 2) *
CInt(Me.txtTransport.Text), 0)
                    afstdLijn = afstdLijn &
CStr(eucl) & " "
                Loop              ' einde fac-
coord-loop
                Close #3        ' sluit, zodat het
daarna heropend kan worden
                Print #1, afstdLijn
                afstdLijn = ""
          Loop                  ' einde
klantencoord.
       Close #2                 ' sluit bestandje 2
       Close #1                 ' sluit bestandje 1

   ' De afstanden zijn opgeslagen, bien

   """""""""""""""""""""""""""
   ' outputbestand vervolledigen '
   """""""""""""""""""""""""""

   strFO = Me.txtOutput(1).Text '  output file
voor memfacloc
   strFO2 = Me.txtOutput(0).Text ' temp file
met afstanden
   strFO3 = Me.txtInput(0).Text ' bestandje
met de vraag

       Open strFO For Append As #1
       Open strFO2 For Input As #2
       Open strFO3 For Input As #3
          Line Input #3, strBuff  ' 1e regel met
header mag weg

          Do Until EOF(3)     ' heel 't vraag bestand
doorlopen
             Line Input #3, strBuff
             Print #1, strBuff       ' de vraag printen
             Line Input #2, strBuff
             Print #1, strBuff       ' de afstand printen
          Loop

       Close #3
       Close #2
       Close #1

   MsgBox "De berekening zijn correct
uitgevoerd!", vbOKOnly, "Whiii"
Exit Sub

ErrRoutine:
   MsgBox "Er is iets mis..." & vbCrLf _
   & "Fout nr " & Err.Number & ": " &
Err.Description, vbExclamation, "Error"

End Sub
```

III Lijst van beste oplossingen voor alle sturingsparameter-combinaties per dataset volgens het memetisch algoritme

Voor elk van de twaalf datasets wordt de beste oplossingen weergegeven. Indien ongeacht welke replicatie en ongeacht welke sturingsparametercombinatie telkens dezelfde beste oplossing tevoorschijn kwam, wordt deze slechts eenmaal weergegeven. Indien dit niet zo is, kan uit het tweede deel van de bestandsnaam gemakkelijk afgeleid worden om welke parametercombinatie het gaat (p = populatiegrootte, g = aantal generaties, l = *local search*). Indien het de standaardparameterwaarden betreft, staat dit niet gegeven. Het laatste deel geeft aan welke replicatie het is.

Klein-geclusterd-variërend

1 2 3 5 7 10 11 12 161129

Klein-willekeurig-variërend

1 2 3 4 5 8 9 10 11 12 142881

Middel-geclusterd-variërend

1 8 10 12 14 16 19 20 21 27 28 31 35 40 456653

Middel-willekeurig-variërend

Bestand: miwil30_001.txt
2 6 8 13 14 15 17 18 19 20 25 28 33 34 36 38 401195
Bestand: miwil30_002.txt
1 2 6 13 14 15 17 18 19 20 25 28 33 34 36 38 401301
Bestand: miwil30_003.txt
2 6 8 13 14 17 18 19 20 25 27 28 34 36 37 38 401245
Bestand: miwil30_004.txt
2 5 6 8 12 13 14 17 18 19 25 27 28 34 36 37 38 401688
Bestand: miwil30_005.txt
1 2 6 13 14 15 17 18 19 20 25 27 28 34 36 38 400180
Bestand: miwil30_g40l8_001.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_g40l8_002.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_g40l8_003.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074

Bestand: miwil30_g40l8_004.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_g40l8_005.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_g40_001.txt
1 2 6 14 15 17 18 20 22 24 25 27 28 34 36 38 405813
Bestand: miwil30_g40_002.txt
2 6 8 13 14 15 17 18 20 22 25 28 33 34 36 38 403126
Bestand: miwil30_g40_003.txt
1 2 6 13 14 15 17 18 19 20 25 27 28 34 36 38 400180
Bestand: miwil30_g40_004.txt
2 5 6 8 12 13 14 15 17 18 19 25 27 28 34 36 38 400517
Bestand: miwil30_g40_005.txt
2 6 8 13 14 15 17 18 19 20 25 28 33 34 36 38 401195
Bestand: miwil30_l8_001.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_l8_002.txt
2 5 6 8 12 13 14 15 17 18 19 25 27 28 34 36 38 400517
Bestand: miwil30_l8_003.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_l8_004.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_l8_005.txt
2 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 38 400509
Bestand: miwil30_p40g40l8_001.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_p40g40l8_002.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_p40g40l8_003.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_p40g40l8_004.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_p40g40l8_005.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_p40g40_001.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_p40g40_002.txt
2 5 6 8 12 13 14 15 17 18 22 25 27 28 34 36 38 402267
Bestand: miwil30_p40g40_003.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_p40g40_004.txt
2 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 38 400509
Bestand: miwil30_p40g40_005.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_p40l8_001.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074

Bestand: miwil30_p40l8_002.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_p40l8_003.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_p40l8_004.txt
1 2 6 13 14 15 17 18 19 20 25 27 28 34 36 38 400180
Bestand: miwil30_p40l8_005.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_p40_001.txt
2 6 8 13 14 17 18 19 20 25 27 28 34 36 37 38 401245
Bestand: miwil30_p40_002.txt
1 2 6 12 13 14 15 17 18 19 25 28 33 34 35 36 38 400615
Bestand: miwil30_p40_003.txt
1 2 6 13 14 15 17 18 19 20 25 27 28 34 36 38 400180
Bestand: miwil30_p40_004.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074
Bestand: miwil30_p40_005.txt
2 6 8 13 14 15 17 18 19 20 25 27 28 34 36 38 400074

Groot-geclusterd-variërend

1 2 3 7 8 9 12 13 15 16 17 18 20 21 22 23 24 25 26 27 28 30 31 32 34 35 37 38 39 40 41 42 45 46 48 3647243

Groot-willekeurig-variërend

1 2 3 4 5 7 8 9 10 13 14 15 17 19 20 22 23 24 25 26 27 28 29 30 31 33 34 35 36 37 38 39 40 41 42 43 44 46 47 49 2888069

Klein-geclusterd-identiek

1 2 3 5 7 10 11 12 163203

Klein-willekeurig-identiek

1 2 3 4 5 8 9 10 11 12 134846

Middel-geclusterd-identiek

1 8 10 12 13 14 16 20 21 22 27 28 31 33 35 40 490493

Middel-willekeurig-identiek

Bestand: mmwil30_001.txt

1 2 3 6 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 381170
Bestand: mmwil30_002.txt
2 6 8 12 13 14 15 17 18 19 25 28 29 33 34 35 36 37 38 380603
Bestand: mmwil30_003.txt
2 3 6 8 12 14 17 18 19 24 25 28 33 34 35 36 37 38 382074
Bestand: mmwil30_004.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_005.txt
2 6 8 12 13 14 15 17 18 19 25 28 29 33 34 35 36 37 38 380603
Bestand: mmwil30_g40_001.txt
1 2 3 6 12 14 17 18 19 24 25 28 33 34 35 36 37 38 382648
Bestand: mmwil30_g40_002.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_g40_003.txt
2 3 6 8 12 14 17 18 19 24 25 28 33 34 35 36 37 38 382074
Bestand: mmwil30_g40_004.txt
2 3 5 6 8 12 13 14 17 18 19 25 26 28 33 34 36 37 38 384407
Bestand: mmwil30_g40_005.txt
2 3 6 8 12 14 17 18 19 24 25 28 33 34 35 36 37 38 382074
Bestand: mmwil30_g40l8_001.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_g40l8_002.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_g40l8_003.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_g40l8_004.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_g40l8_005.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_l8_001.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_l8_002.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_l8_003.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_l8_004.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_l8_005.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40_001.txt
2 3 6 8 12 14 17 18 19 24 25 28 33 34 35 36 37 38 382074
Bestand: mmwil30_p40_002.txt
2 3 6 8 12 14 15 17 18 19 24 25 28 33 34 35 36 38 382456
Bestand: mmwil30_p40_003.txt
2 6 8 12 13 14 15 17 18 19 25 28 29 33 34 35 36 37 38 380603
Bestand: mmwil30_p40_004.txt

2 3 6 8 12 14 15 17 18 19 24 25 28 33 34 35 36 38 382456
Bestand: mmwil30_p40_005.txt
2 3 6 8 12 14 17 18 19 24 25 28 33 34 35 36 37 38 382074
Bestand: mmwil30_p40g40_001.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40g40_002.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40g40_003.txt
2 3 6 8 12 14 17 18 19 24 25 28 33 34 35 36 37 38 382074
Bestand: mmwil30_p40g40_004.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40g40_005.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40g40l8_001.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40g40l8_002.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40g40l8_003.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40g40l8_004.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40g40l8_005.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40l8_001.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40l8_002.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40l8_003.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40l8_004.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596
Bestand: mmwil30_p40l8_005.txt
2 3 6 8 12 13 14 15 17 18 19 25 28 33 34 35 36 37 38 380596

Groot-geclusterd-identiek

1 2 3 4 7 8 9 12 13 14 15 16 17 18 20 21 22 23 24 25 26 27 28 30 31 32 33 34 35 37 38 39 40 41 42 45 46 48 50 3732750

Groot-willekeurig-identiek

1 2 3 4 5 7 8 9 10 13 14 15 18 19 20 22 23 24 25 26 27 28 29 30 31 34 35 36 37 38 39 40 41 42 43 44 46 47 48 49 50 2848717

IV Klantengegevens voor het onderzoek

X	x-coördinaat
Y	y-coördinaat
V	vraag

Kleine onderneming

	geclusterd		willekeurig				geclusterd		willekeurig				geclusterd		willekeurig		
	X	Y	X	Y	V		X	Y	X	Y	V		X	Y	X	Y	V
1	16	20	7	2	13	*10*	91	68	20	1	6	*18*	52	51	14	98	6
2	11	22	34	5	18	*11*	94	75	71	34	11	*19*	64	52	72	65	11
3	2	26	86	27	12	*12*	84	77	58	37	12	*20*	52	56	18	49	6
4	6	18	4	56	3	*13*	87	77	57	31	7	*21*	21	80	83	44	5
5	11	8	49	18	4	*14*	86	70	1	24	4	*22*	20	91	50	40	15
6	8	18	79	43	6	*15*	59	57	33	94	9	*23*	18	89	1	5	8
7	7	25	93	70	14	*16*	48	52	90	45	6	*24*	23	65	49	95	13
8	7	15	41	25	5	*17*	69	41	25	57	12	*25*	20	72	72	16	6
9	6	20	94	61	14												

Middelgrote onderneming

	geclusterd		willekeurig				geclusterd		willekeurig				geclusterd		willekeurig		
	X	Y	X	Y	V		X	Y	X	Y	V		X	Y	X	Y	V
1	13	11	137	62	3	*35*	9	10	90	102	2	*68*	56	120	23	71	6
2	10	13	11	110	7	*36*	12	6	58	122	12	*69*	118	26	2	14	14
3	13	12	35	125	13	*37*	11	16	103	93	8	*70*	108	22	72	22	15
4	1	16	59	14	10	*38*	9	13	3	68	14	*71*	113	40	94	74	1
5	7	6	110	57	2	*39*	12	23	47	64	14	*72*	106	44	7	112	3
6	9	12	71	23	2	*40*	103	136	15	64	10	*73*	114	31	127	123	12
7	13	17	103	120	14	*41*	100	134	45	85	4	*74*	116	54	53	60	13
8	12	14	133	97	9	*42*	97	133	12	63	5	*75*	111	48	106	39	1
9	0	12	16	128	1	*43*	100	135	36	134	2	*76*	102	41	37	14	3
10	2	15	128	7	5	*44*	102	136	136	98	6	*77*	121	31	138	140	1
11	13	14	125	23	2	*45*	95	128	56	134	5	*78*	13	87	62	16	7
12	18	12	19	73	2	*46*	94	128	22	97	11	*79*	19	79	15	115	2
13	3	20	76	140	7	*47*	98	135	21	9	6	*80*	15	70	7	17	4
14	18	9	67	52	6	*48*	99	133	85	54	3	*81*	19	86	91	32	5
15	17	19	83	16	15	*49*	97	133	41	132	3	*82*	26	86	28	67	2
16	9	18	47	78	13	*50*	61	116	116	116	13	*83*	20	78	124	107	4
17	8	16	16	68	10	*51*	52	124	94	81	13	*84*	17	80	54	77	14
18	11	8	65	40	4	*52*	69	109	70	117	6	*85*	25	66	44	106	2
19	13	17	16	41	3	*53*	60	118	47	43	13	*86*	16	71	131	85	14
20	10	5	66	63	9	*54*	68	109	78	123	15	*87*	23	76	19	79	7
21	6	14	86	121	2	*55*	60	129	75	25	2	*88*	23	81	100	110	4
22	10	14	107	11	6	*56*	64	109	135	87	11	*89*	22	78	62	131	14

23	14	15	9	48	13	*57*	56	117	77	132	14	*90*	17	71	31	127	10
24	13	14	57	70	8	*58*	62	121	106	21	8	*91*	19	78	65	45	14
25	17	17	127	28	15	*59*	61	102	104	42	11	*92*	19	87	1	107	1
26	10	12	66	117	9	*60*	58	108	52	65	1	*93*	20	81	58	134	5
27	19	16	128	114	15	*61*	49	113	96	77	14	*94*	20	79	47	49	6
28	0	16	128	64	14	*62*	54	125	92	136	11	*95*	23	66	120	94	9
29	14	18	116	81	8	*63*	62	106	89	78	9	*96*	16	65	62	114	6
30	13	19	40	45	8	*64*	67	116	73	37	5	*97*	32	79	97	110	9
31	7	9	130	20	9	*65*	68	122	17	75	5	*98*	22	78	71	75	5
32	12	13	75	123	12	*66*	63	108	120	12	6	*99*	31	77	40	103	7
33	13	13	121	14	1	*67*	49	117	66	55	7	*100*	25	76	11	131	6
34	8	20	17	4	5												

Grote onderneming

	geclusterd		willekeurig				geclusterd		willekeurig				geclusterd		willekeurig		
	X	Y	X	Y	V		X	Y	X	Y	V		X	Y	X	Y	V
1	100	53	26	161	6	*168*	100	210	157	192	12	*335*	226	39	137	112	5
2	93	61	148	57	7	*169*	99	223	18	193	2	*336*	227	38	187	27	1
3	106	57	106	201	7	*170*	96	221	6	59	5	*337*	209	32	223	139	7
4	92	51	44	199	12	*171*	26	74	217	152	11	*338*	157	196	54	51	12
5	100	46	16	134	11	*172*	27	68	139	2	6	*339*	153	194	48	218	7
6	100	53	35	21	10	*173*	25	74	4	18	12	*340*	154	197	192	110	15
7	105	49	239	115	1	*174*	34	71	62	23	2	*341*	149	194	221	82	6
8	101	55	155	118	15	*175*	18	72	129	171	1	*342*	149	198	87	142	11
9	100	55	124	134	4	*176*	26	79	60	57	11	*343*	151	187	213	155	7
10	92	53	65	77	11	*177*	27	74	215	8	10	*344*	154	199	204	183	7
11	103	47	221	133	13	*178*	26	73	176	149	2	*345*	142	189	117	249	14
12	102	55	89	4	15	*179*	31	79	187	241	5	*346*	153	193	170	62	10
13	103	45	160	63	8	*180*	28	78	243	26	10	*347*	150	194	242	13	4
14	105	64	113	199	1	*181*	22	72	139	213	8	*348*	159	195	148	179	1
15	101	55	213	69	10	*182*	22	79	72	39	1	*349*	155	195	36	81	4
16	102	52	182	64	13	*183*	33	73	231	229	10	*350*	153	188	134	61	3
17	109	57	172	241	10	*184*	30	83	188	81	13	*351*	152	195	180	132	3
18	109	59	234	77	15	*185*	27	65	132	56	6	*352*	145	193	216	223	12
19	100	55	186	88	7	*186*	26	75	85	58	14	*353*	153	194	7	107	14
20	106	61	15	84	8	*187*	28	77	131	13	1	*354*	152	193	126	139	11
21	105	58	186	192	3	*188*	29	69	42	70	4	*355*	157	199	244	126	12
22	103	56	40	74	6	*189*	32	81	175	96	11	*356*	159	189	25	81	4
23	101	53	79	100	2	*190*	21	81	9	24	10	*357*	151	192	52	237	8
24	92	53	250	95	3	*191*	35	69	95	78	6	*358*	152	193	176	222	3
25	93	53	102	200	4	*192*	27	80	22	89	8	*359*	147	192	98	72	10
26	97	50	64	220	15	*193*	23	77	99	171	5	*360*	159	199	234	181	5
27	97	58	202	41	10	*194*	27	75	225	248	13	*361*	157	193	102	127	5
28	103	54	206	158	11	*195*	19	76	151	99	10	*362*	146	194	226	64	15
29	101	55	6	40	12	*196*	26	71	132	231	11	*363*	145	191	175	66	10
30	99	53	144	226	12	*197*	27	71	21	198	8	*364*	152	186	192	122	14
31	93	51	94	242	10	*198*	30	75	21	164	13	*365*	157	186	136	26	4

32	95	54	170	77	3	*199*	24	77	143	183	5	*366*	153	195	124	124	1
33	108	49	200	246	14	*200*	27	81	150	128	15	*367*	150	195	54	53	8
34	110	56	24	165	4	*201*	22	72	24	39	2	*368*	153	186	111	235	2
35	95	51	232	212	12	*202*	33	68	124	188	3	*369*	152	194	12	179	15
36	96	54	148	122	14	*203*	27	75	63	220	9	*370*	158	194	245	66	10
37	97	56	226	190	6	*204*	139	125	57	56	8	*371*	150	197	90	242	14
38	104	54	245	23	1	*205*	143	123	33	122	11	*372*	148	187	226	110	7
39	98	53	53	65	2	*206*	132	133	144	150	15	*373*	141	194	214	174	10
40	110	52	134	64	13	*207*	126	114	14	238	10	*374*	153	197	222	188	3
41	100	57	224	238	13	*208*	137	122	59	222	7	*375*	155	192	145	211	1
42	101	55	120	211	14	*209*	150	125	14	130	3	*376*	154	193	203	165	15
43	97	64	237	60	1	*210*	128	126	154	208	15	*377*	153	190	82	22	7
44	106	48	56	241	14	*211*	140	122	221	75	11	*378*	156	197	31	217	15
45	97	52	101	87	13	*212*	136	123	32	33	9	*379*	160	187	143	189	3
46	25	149	150	225	11	*213*	137	125	146	135	5	*380*	154	190	191	169	4
47	22	148	250	207	2	*214*	134	111	58	169	2	*381*	153	195	159	147	9
48	22	148	52	216	2	*215*	140	135	142	21	6	*382*	156	191	216	216	9
49	25	148	172	9	8	*216*	141	123	142	185	15	*383*	147	196	88	216	15
50	21	147	99	54	8	*217*	130	126	144	131	4	*384*	147	195	243	29	5
51	29	145	47	107	4	*218*	144	122	158	91	10	*385*	237	211	223	19	14
52	28	142	1	9	1	*219*	141	121	135	185	2	*386*	237	207	69	105	12
53	25	148	211	227	12	*220*	150	116	55	91	11	*387*	226	196	195	116	9
54	24	153	169	8	12	*221*	135	126	25	145	8	*388*	238	198	36	197	5
55	25	141	225	33	4	*222*	127	128	228	168	8	*389*	235	206	45	240	10
56	24	143	189	154	8	*223*	136	123	44	56	2	*390*	236	209	21	83	2
57	24	150	177	172	11	*224*	137	123	47	123	14	*391*	233	199	10	40	13
58	21	153	188	130	11	*225*	143	140	199	110	15	*392*	234	196	229	119	12
59	25	149	180	62	2	*226*	145	112	36	174	15	*393*	243	191	208	216	3
60	24	148	173	161	2	*227*	138	123	227	76	6	*394*	241	202	17	5	14
61	23	147	216	228	15	*228*	142	134	144	213	7	*395*	238	200	46	64	3
62	27	147	182	44	5	*229*	128	134	19	87	2	*396*	242	205	159	6	8
63	29	141	98	170	12	*230*	129	124	95	109	4	*397*	236	193	127	211	10
64	26	147	192	34	11	*231*	140	114	7	234	10	*398*	244	207	73	77	7
65	23	149	189	78	1	*232*	133	122	130	101	11	*399*	247	204	124	74	10
66	30	144	127	200	5	*233*	143	126	141	123	9	*400*	240	188	98	187	3
67	24	151	243	40	6	*234*	130	120	70	170	10	*401*	238	194	232	154	3
68	19	145	154	172	12	*235*	148	128	118	116	13	*402*	232	192	172	202	4
69	23	148	148	85	3	*236*	149	114	129	235	4	*403*	230	204	41	147	6
70	27	141	163	219	4	*237*	146	114	173	122	12	*404*	236	200	95	135	15
71	26	148	49	209	11	*238*	136	124	209	29	8	*405*	238	197	64	218	3
72	17	145	171	193	7	*239*	128	116	128	13	1	*406*	241	196	156	68	6
73	25	150	47	115	15	*240*	122	127	140	208	15	*407*	232	197	192	103	9
74	26	144	109	64	15	*241*	144	125	24	164	2	*408*	228	198	174	241	8
75	23	147	162	54	15	*242*	137	125	105	158	5	*409*	244	209	116	88	2
76	26	149	27	113	6	*243*	124	134	223	161	13	*410*	235	191	37	230	7
77	17	147	137	214	12	*244*	135	139	47	100	10	*411*	237	199	58	134	5
78	29	143	154	133	9	*245*	125	120	11	151	12	*412*	241	191	36	233	9
79	32	150	18	50	11	*246*	140	115	72	213	4	*413*	243	195	84	13	9

80	28	152	216	88	6	*247*	139	118	161	230	10	*414*	234	199	51	129	7
81	28	143	9	199	12	*248*	124	113	69	87	14	*415*	234	199	137	161	7
82	26	155	157	30	15	*249*	135	124	86	9	4	*416*	231	209	36	189	3
83	28	153	166	150	9	*250*	133	123	137	91	4	*417*	245	198	84	64	5
84	25	150	124	68	10	*251*	136	121	131	113	10	*418*	230	195	58	227	4
85	19	153	50	107	11	*252*	150	128	239	64	13	*419*	231	207	60	177	3
86	25	151	171	133	15	*253*	136	123	111	67	12	*420*	241	191	64	107	3
87	22	150	58	81	9	*254*	134	126	152	163	4	*421*	234	198	90	104	2
88	20	142	29	59	1	*255*	151	124	174	127	9	*422*	235	203	121	39	11
89	21	149	8	98	15	*256*	142	125	48	107	8	*423*	230	195	144	17	2
90	21	148	87	194	5	*257*	132	128	47	236	10	*424*	232	192	160	43	6
91	26	146	90	155	11	*258*	138	125	34	243	15	*425*	248	199	20	212	2
92	22	141	138	199	7	*259*	134	138	162	123	2	*426*	234	202	50	201	12
93	27	150	208	119	6	*260*	131	139	210	88	13	*427*	238	202	154	214	15
94	25	148	109	70	12	*261*	135	135	140	77	4	*428*	246	198	29	159	3
95	26	148	229	167	13	*262*	147	123	59	160	10	*429*	233	198	190	166	15
96	27	149	127	60	11	*263*	136	124	70	223	14	*430*	233	195	3	184	9
97	24	146	31	202	14	*264*	138	124	139	192	13	*431*	230	194	55	24	2
98	30	148	169	88	6	*265*	128	120	75	85	3	*432*	249	201	191	42	9
99	21	154	89	248	3	*266*	132	121	149	35	10	*433*	233	205	192	250	12
100	27	144	250	230	7	*267*	142	115	170	234	9	*434*	238	196	167	213	3
101	23	148	55	235	9	*268*	135	134	170	99	4	*435*	232	195	201	8	13
102	26	151	29	64	9	*269*	136	125	63	247	2	*436*	241	195	49	230	9
103	96	220	62	90	10	*270*	138	114	24	169	12	*437*	247	202	115	218	8
104	104	217	25	21	6	*271*	132	128	65	225	13	*438*	231	199	172	191	15
105	95	220	226	56	9	*272*	131	123	228	77	13	*439*	209	82	204	28	11
106	100	219	69	32	9	*273*	138	121	184	36	9	*440*	203	79	1	247	7
107	107	221	92	201	7	*274*	123	121	8	247	5	*441*	198	92	123	141	2
108	99	219	7	25	9	*275*	220	38	242	100	4	*442*	210	85	197	122	2
109	99	218	166	116	8	*276*	210	24	2	230	7	*443*	204	98	94	194	13
110	98	211	157	196	7	*277*	217	38	220	187	9	*444*	199	87	240	74	5
111	101	217	215	172	6	*278*	216	38	55	43	5	*445*	208	98	16	18	13
112	100	223	198	235	10	*279*	225	50	174	77	1	*446*	216	86	224	217	5
113	95	222	96	142	7	*280*	224	31	192	208	8	*447*	207	92	105	22	15
114	99	210	16	60	8	*281*	210	42	19	231	11	*448*	210	92	120	230	4
115	94	218	47	13	7	*282*	214	29	48	132	13	*449*	203	86	14	75	14
116	105	216	159	17	12	*283*	213	30	19	22	13	*450*	210	83	52	51	4
117	97	219	143	156	6	*284*	201	39	216	12	4	*451*	209	86	120	89	1
118	100	216	37	1	13	*285*	220	26	92	19	15	*452*	215	97	219	155	7
119	103	214	175	133	8	*286*	209	42	246	221	9	*453*	216	85	178	137	12
120	95	215	183	209	8	*287*	219	34	189	128	1	*454*	209	85	5	119	2
121	105	216	208	12	15	*288*	218	38	12	110	9	*455*	212	89	176	156	5
122	92	221	77	130	11	*289*	210	44	146	16	14	*456*	209	80	144	130	9
123	95	221	43	27	4	*290*	218	38	202	120	1	*457*	213	96	183	62	6
124	107	220	148	25	6	*291*	220	42	152	174	6	*458*	209	83	133	41	2
125	99	218	180	167	15	*292*	204	38	65	227	10	*459*	206	95	115	180	13
126	97	217	176	79	13	*293*	214	47	73	57	3	*460*	207	84	87	74	10
127	101	212	173	119	7	*294*	210	29	250	29	14	*461*	210	84	78	211	5

128	98	217	202	242	5	*295*	216	32	158	245	8	*462*	213	98	87	36	12
129	101	220	145	89	8	*296*	215	36	144	4	6	*463*	216	94	61	190	3
130	95	211	76	242	7	*297*	223	34	250	51	3	*464*	215	91	35	199	9
131	101	221	90	47	13	*298*	206	35	234	180	5	*465*	210	88	164	209	15
132	98	218	144	115	15	*299*	215	24	15	105	4	*466*	207	81	242	109	2
133	101	216	106	181	2	*300*	212	45	144	153	9	*467*	209	86	105	186	15
134	100	214	217	40	7	*301*	219	32	163	68	5	*468*	214	95	169	42	13
135	99	218	149	135	7	*302*	229	33	37	222	1	*469*	208	96	7	155	7
136	100	214	158	205	12	*303*	216	38	244	55	3	*470*	211	86	111	94	9
137	99	226	14	138	9	*304*	218	44	211	93	9	*471*	203	85	133	227	1
138	99	222	157	100	4	*305*	215	25	140	190	14	*472*	215	80	18	8	11
139	105	213	68	99	10	*306*	217	39	102	225	10	*473*	204	74	108	138	15
140	101	218	26	216	14	*307*	215	38	39	17	8	*474*	209	82	132	88	4
141	100	217	119	241	8	*308*	216	44	211	190	10	*475*	210	80	162	224	11
142	100	219	47	116	5	*309*	214	36	145	25	11	*476*	197	89	223	126	8
143	100	222	238	67	10	*310*	216	34	117	45	10	*477*	209	81	104	229	6
144	100	217	151	137	6	*311*	217	38	122	122	4	*478*	209	85	49	173	1
145	96	215	70	193	12	*312*	226	29	76	163	13	*479*	208	75	137	38	15
146	101	221	241	125	7	*313*	216	53	8	74	15	*480*	205	84	169	66	15
147	103	214	213	130	6	*314*	214	44	81	214	8	*481*	210	95	189	33	14
148	104	220	56	19	3	*315*	214	40	34	181	8	*482*	204	89	114	176	15
149	102	214	9	246	6	*316*	214	39	81	25	9	*483*	205	85	162	197	13
150	101	216	213	103	3	*317*	206	39	56	205	5	*484*	219	84	189	135	11
151	100	215	169	205	11	*318*	205	45	2	117	6	*485*	206	95	84	118	4
152	100	213	219	243	13	*319*	224	50	228	230	4	*486*	210	91	52	211	7
153	97	214	189	82	7	*320*	213	36	102	112	9	*487*	200	86	188	28	15
154	100	221	71	136	2	*321*	202	32	117	27	2	*488*	208	76	115	223	1
155	97	212	18	104	3	*322*	210	42	15	162	5	*489*	207	97	94	16	9
156	92	221	67	102	14	*323*	228	45	170	80	8	*490*	199	80	82	18	6
157	103	217	208	195	8	*324*	218	35	244	145	14	*491*	210	86	101	126	11
158	101	222	93	9	7	*325*	210	46	31	72	2	*492*	204	84	25	2	3
159	103	214	115	13	1	*326*	217	38	148	119	6	*493*	211	87	176	136	12
160	99	216	95	185	8	*327*	216	38	204	40	11	*494*	198	81	20	12	15
161	104	215	113	212	9	*328*	216	43	191	239	10	*495*	206	97	92	140	12
162	104	222	23	86	9	*329*	212	43	68	147	8	*496*	220	81	222	107	6
163	96	215	29	36	12	*330*	213	35	69	198	2	*497*	211	88	26	25	5
164	101	217	219	101	3	*331*	213	30	56	91	3	*498*	206	86	142	240	11
165	102	211	191	215	3	*332*	216	35	182	142	12	*499*	212	81	230	104	3
166	99	217	84	55	15	*333*	213	38	202	1	8	*500*	208	86	94	208	7
167	100	218	150	183	6	*334*	225	50	16	221	11						

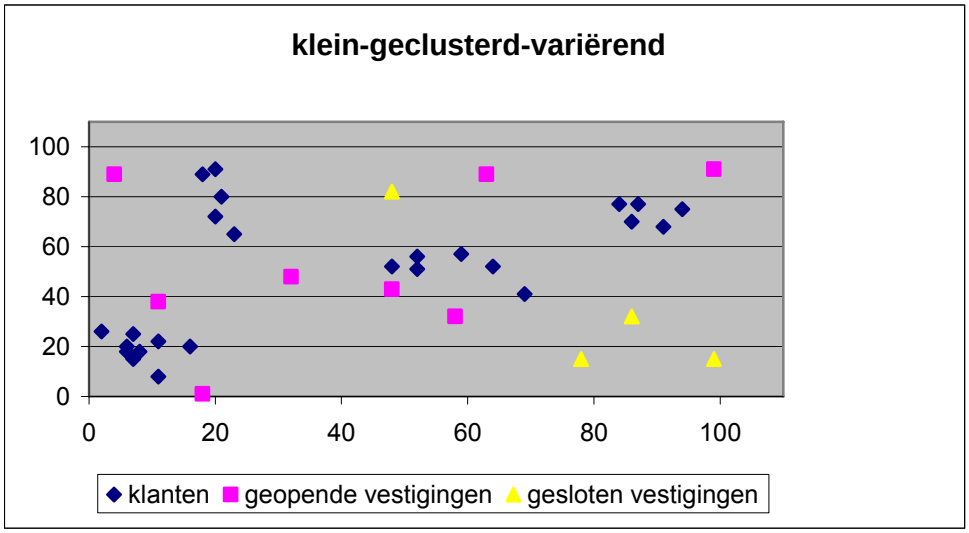
V Vestigingengegevens en de resultaten

In totaal hebben we 12 dataset waarvoor hieronder telkens de beste oplossing wordt weergegeven.

	De vestiging is voor deze dataset geopend
	De vestiging is voor deze dataset gesloten

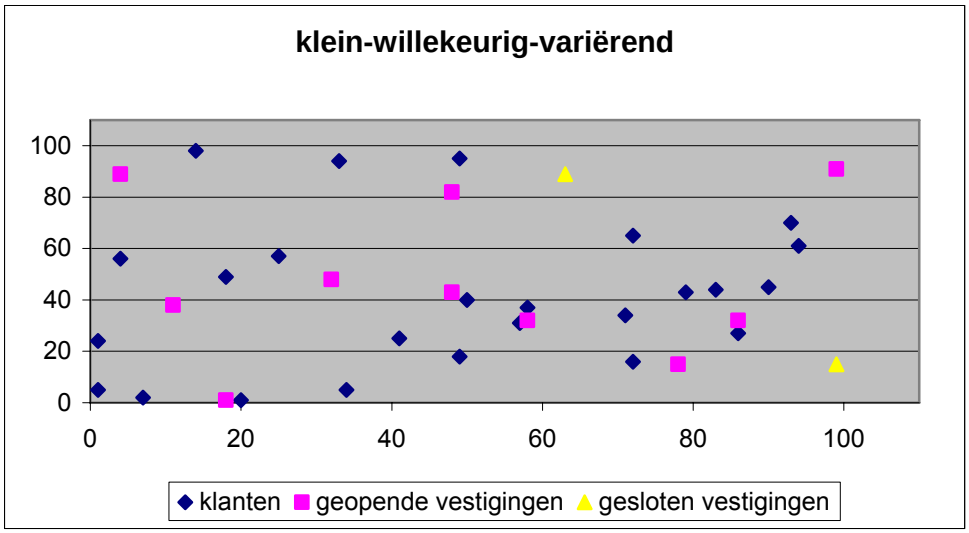
Klein-geclusterd-variërend

	X	Y	capaciteit	kost		X	Y	capaciteit	kost
1	58	32	20	1755	7	63	89	37	3294
2	48	43	23	2102	8	48	82	28	2853
3	32	48	32	3455	9	78	15	21	2194
4	86	32	27	3372	10	11	38	37	4654
5	18	1	27	3202	11	99	91	25	2631
6	99	15	34	2823	12	4	89	40	4823



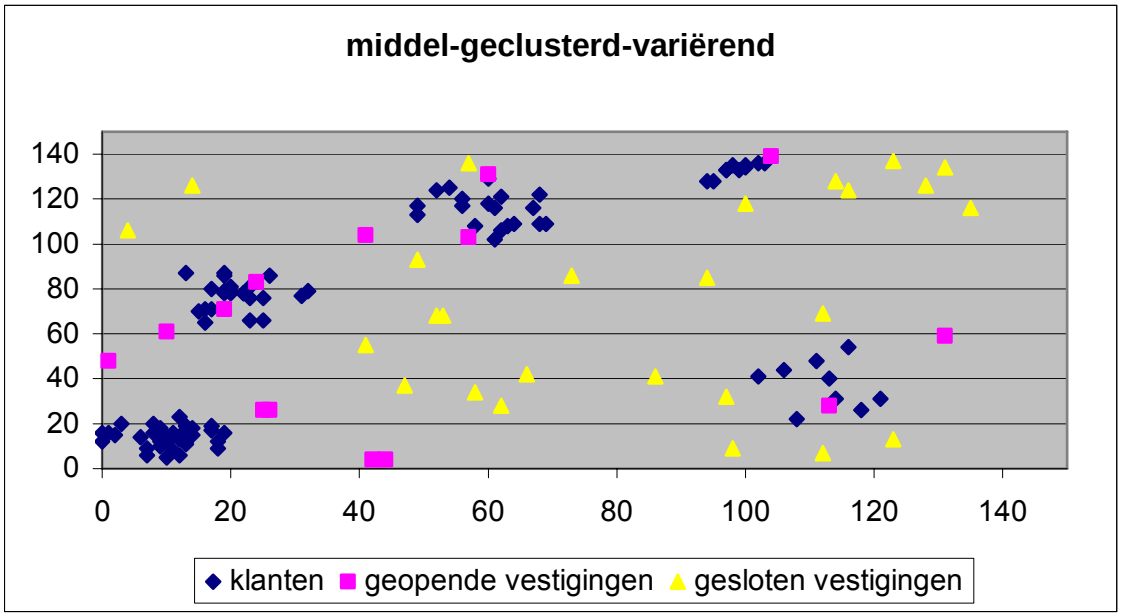
Klein-willekeurig-variërend

	X	Y	capaciteit	kost		X	Y	capaciteit	kost
1	58	32	20	1755	7	63	89	37	3294
2	48	43	23	2102	8	48	82	28	2853
3	32	48	32	3455	9	78	15	21	2194
4	86	32	27	3372	10	11	38	37	4654
5	18	1	27	3202	11	99	91	25	2631
6	99	15	34	2823	12	4	89	40	4823



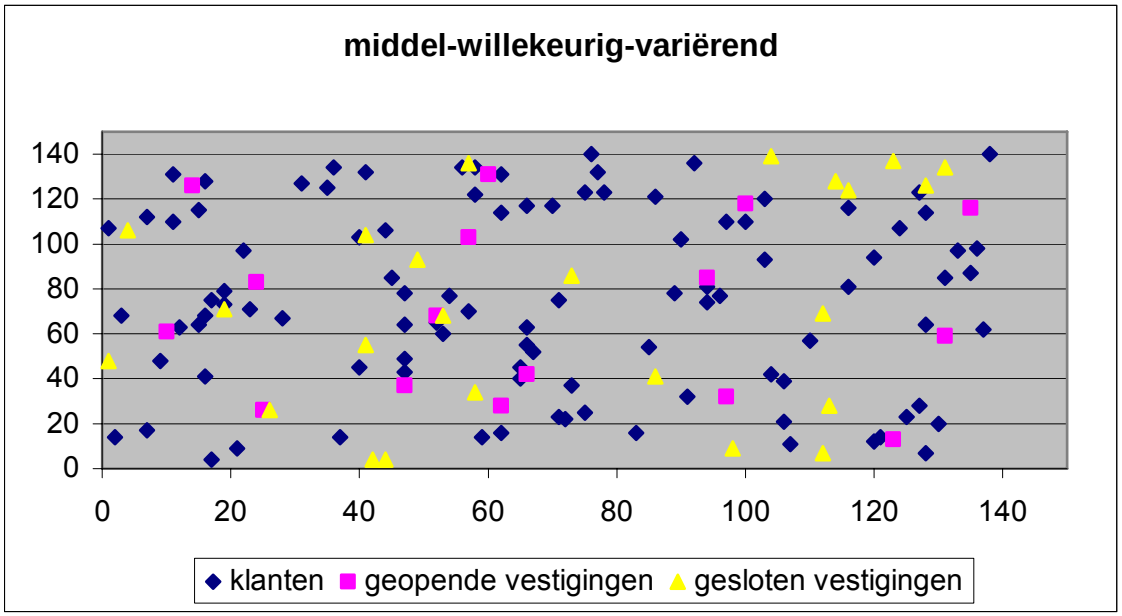
Middel-geclusterd-variërend

	X	Y	capaciteit	kost		X	Y	capaciteit	kost
1	26	26	64	7266	21	113	28	48	5560
2	94	85	42	3984	22	112	69	79	9100
3	116	124	63	6207	23	112	7	64	8014
4	58	34	47	5172	24	41	55	63	5820
5	4	106	54	4833	25	123	13	76	8128
6	62	28	42	4194	26	49	93	52	5059
7	123	137	57	4693	27	57	103	63	6321
8	25	26	71	7734	28	60	131	64	6796
9	86	41	78	8280	29	128	126	45	4100
10	42	4	43	4287	30	131	134	59	4843
11	73	86	73	7443	31	44	4	54	5182
12	19	71	71	7009	32	98	9	53	5381
13	47	37	44	4049	33	57	136	71	6665
14	10	61	57	6396	34	100	118	71	7399
15	52	68	78	8149	35	41	104	72	7441
16	104	139	50	4667	36	14	126	60	6637
17	66	42	50	6404	37	53	68	78	9484
18	97	32	63	6246	38	135	116	60	5373
19	131	59	71	8830	39	114	128	41	4640
20	24	83	61	5817	40	1	48	58	4975



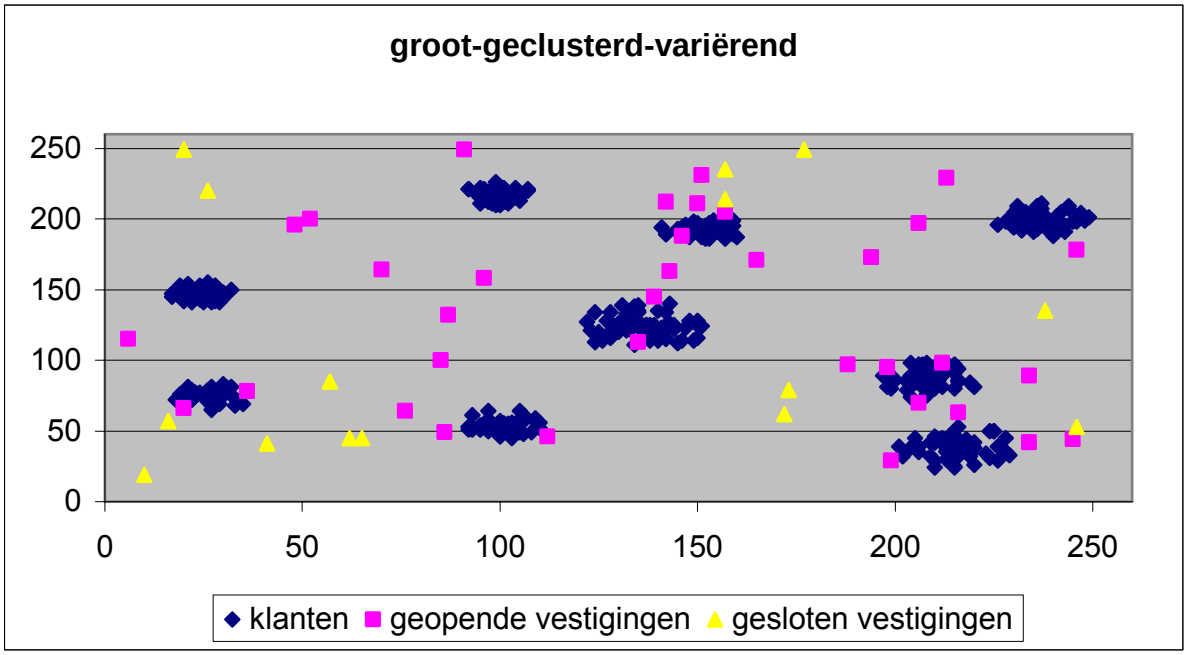
Middel-willekeurig-variërend

	X	Y	capaciteit	kost		X	Y	capaciteit	kost
1	26	26	64	7266	21	113	28	48	5560
2	94	85	42	3984	22	112	69	79	9100
3	116	124	63	6207	23	112	7	64	8014
4	58	34	47	5172	24	41	55	63	5820
5	4	106	54	4833	25	123	13	76	8128
6	62	28	42	4194	26	49	93	52	5059
7	123	137	57	4693	27	57	103	63	6321
8	25	26	71	7734	28	60	131	64	6796
9	86	41	78	8280	29	128	126	45	4100
10	42	4	43	4287	30	131	134	59	4843
11	73	86	73	7443	31	44	4	54	5182
12	19	71	71	7009	32	98	9	53	5381
13	47	37	44	4049	33	57	136	71	6665
14	10	61	57	6396	34	100	118	71	7399
15	52	68	78	8149	35	41	104	72	7441
16	104	139	50	4667	36	14	126	60	6637
17	66	42	50	6404	37	53	68	78	9484
18	97	32	63	6246	38	135	116	60	5373
19	131	59	71	8830	39	114	128	41	4640
20	24	83	61	5817	40	1	48	58	4975



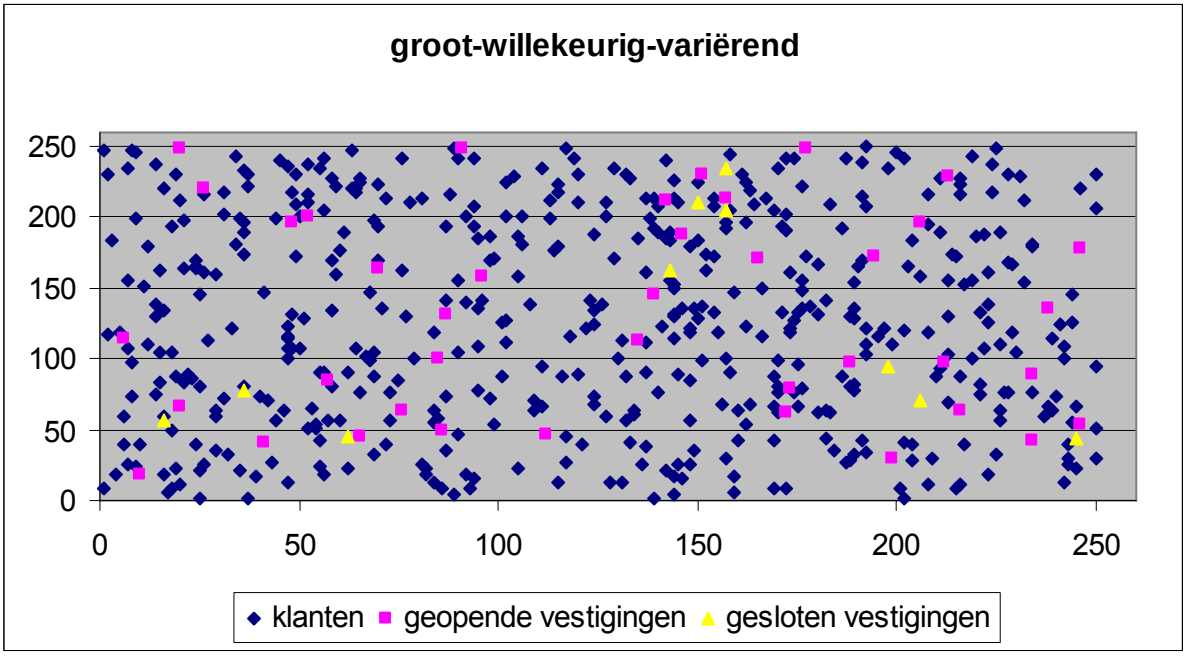
Groot-geclusterd-variërend

	X	Y	capaciteit	kost		X	Y	capaciteit	kost
1	146	188	142	12977	26	216	63	91	8374
2	85	100	139	12403	27	91	249	135	14426
3	87	132	121	14974	28	139	145	134	14488
4	65	45	85	9397	29	10	19	138	13571
5	20	249	97	7945	30	234	89	125	15448
6	62	45	120	15445	31	234	42	113	10484
7	194	173	138	16850	32	143	163	91	9668
8	151	231	128	11482	33	157	214	89	8076
9	135	113	95	9425	34	52	200	102	12950
10	41	41	107	12519	35	96	158	136	16345
11	157	235	105	11246	36	177	249	113	13106
12	245	44	135	13852	37	142	212	143	14689
13	86	49	117	11327	38	70	164	114	9502
14	173	79	89	7933	39	212	98	86	10448
15	206	197	103	8301	40	213	229	140	15311
16	198	95	145	14681	41	76	64	144	18280
17	20	66	146	16258	42	48	196	100	8079
18	157	205	120	14561	43	238	135	133	14302
19	246	53	91	7838	44	26	220	150	17584
20	112	46	134	14318	45	150	211	122	12040
21	206	70	137	12179	46	188	97	135	14108
22	246	178	109	13799	47	172	62	91	10453
23	165	171	138	11202	48	36	78	110	12540
24	6	115	112	10109	49	57	85	110	12298
25	199	29	110	8904	50	16	57	123	15988



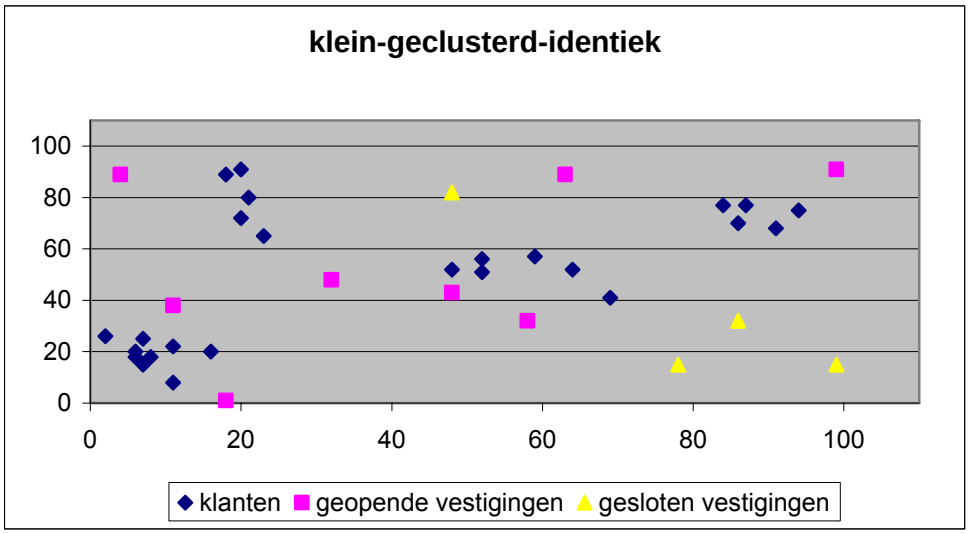
Groot-willekeurig-variërend

	X	Y	capaciteit	kost		X	Y	capaciteit	kost
1	146	188	142	12977	26	216	63	91	8374
2	85	100	139	12403	27	91	249	135	14426
3	87	132	121	14974	28	139	145	134	14488
4	65	45	85	9397	29	10	19	138	13571
5	20	249	97	7945	30	234	89	125	15448
6	62	45	120	15445	31	234	42	113	10484
7	194	173	138	16850	32	143	163	91	9668
8	151	231	128	11482	33	157	214	89	8076
9	135	113	95	9425	34	52	200	102	12950
10	41	41	107	12519	35	96	158	136	16345
11	157	235	105	11246	36	177	249	113	13106
12	245	44	135	13852	37	142	212	143	14689
13	86	49	117	11327	38	70	164	114	9502
14	173	79	89	7933	39	212	98	86	10448
15	206	197	103	8301	40	213	229	140	15311
16	198	95	145	14681	41	76	64	144	18280
17	20	66	146	16258	42	48	196	100	8079
18	157	205	120	14561	43	238	135	133	14302
19	246	53	91	7838	44	26	220	150	17584
20	112	46	134	14318	45	150	211	122	12040
21	206	70	137	12179	46	188	97	135	14108
22	246	178	109	13799	47	172	62	91	10453
23	165	171	138	11202	48	36	78	110	12540
24	6	115	112	10109	49	57	85	110	12298
25	199	29	110	8904	50	16	57	123	15988



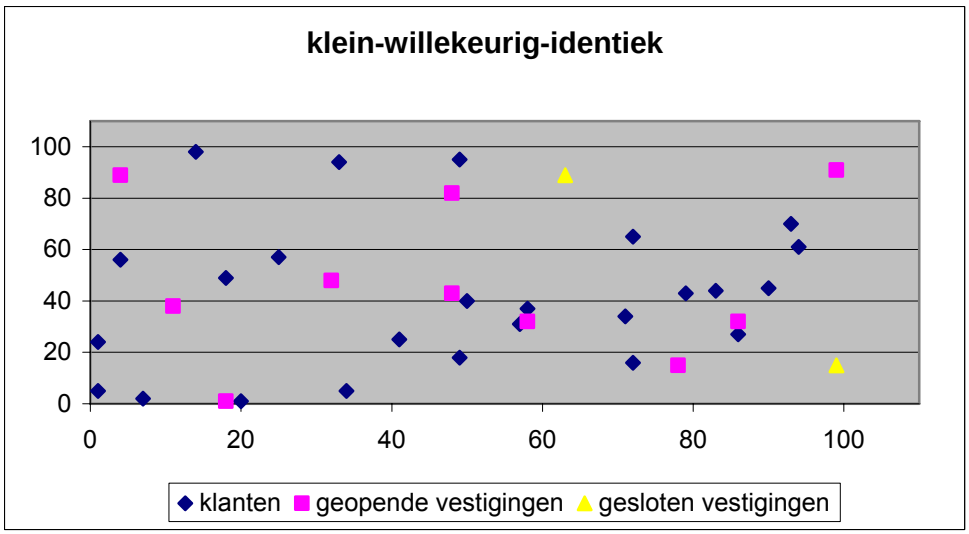
Klein-geclusterd-identiek

	X	Y	capaciteit	kost		X	Y	capaciteit	kost
1	58	32	30	3000	7	63	89	30	3000
2	48	43	30	3000	8	48	82	30	3000
3	32	48	30	3000	9	78	15	30	3000
4	86	32	30	3000	10	11	38	30	3000
5	18	1	30	3000	11	99	91	30	3000
6	99	15	30	3000	12	4	89	30	3000



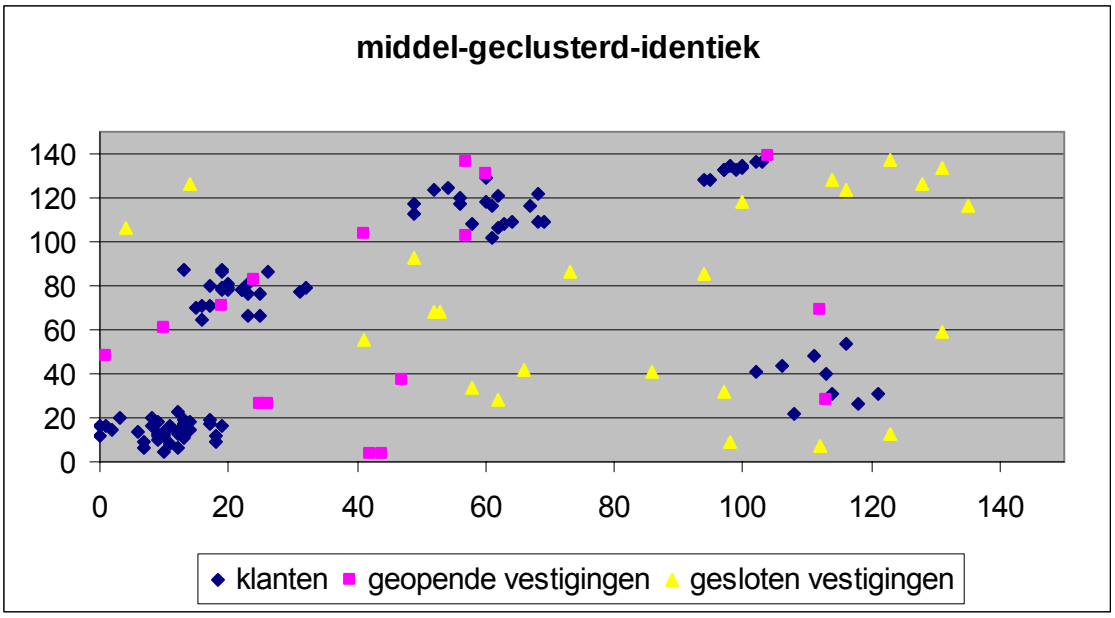
Klein-willekeurig-identiek

	X	Y	capaciteit	kost		X	Y	capaciteit	kost
1	58	32	30	3000	7	63	89	30	3000
2	48	43	30	3000	8	48	82	30	3000
3	32	48	30	3000	9	78	15	30	3000
4	86	32	30	3000	10	11	38	30	3000
5	18	1	30	3000	11	99	91	30	3000
6	99	15	30	3000	12	4	89	30	3000



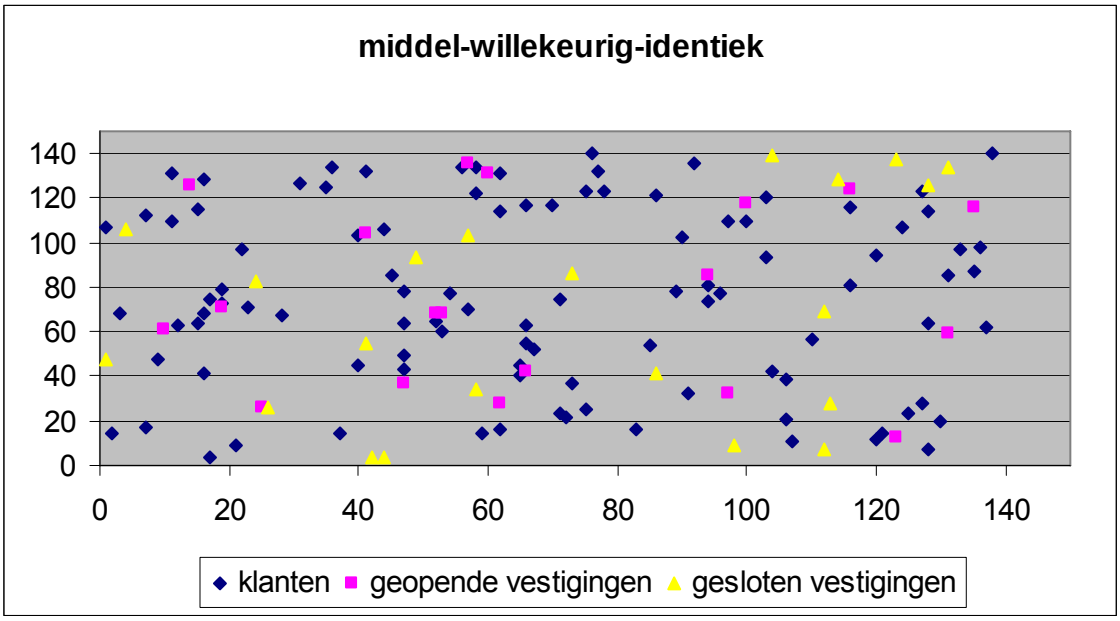
Middel-geclusterd-identiek

	X	Y	capaciteit	kost		X	Y	capaciteit	kost
1	26	26	50	5000	21	113	28	50	5000
2	94	85	50	5000	22	112	69	50	5000
3	116	124	50	5000	23	112	7	50	5000
4	58	34	50	5000	24	41	55	50	5000
5	4	106	50	5000	25	123	13	50	5000
6	62	28	50	5000	26	49	93	50	5000
7	123	137	50	5000	27	57	103	50	5000
8	25	26	50	5000	28	60	131	50	5000
9	86	41	50	5000	29	128	126	50	5000
10	42	4	50	5000	30	131	134	50	5000
11	73	86	50	5000	31	44	4	50	5000
12	19	71	50	5000	32	98	9	50	5000
13	47	37	50	5000	33	57	136	50	5000
14	10	61	50	5000	34	100	118	50	5000
15	52	68	50	5000	35	41	104	50	5000
16	104	139	50	5000	36	14	126	50	5000
17	66	42	50	5000	37	53	68	50	5000
18	97	32	50	5000	38	135	116	50	5000
19	131	59	50	5000	39	114	128	50	5000
20	24	83	50	5000	40	1	48	50	5000



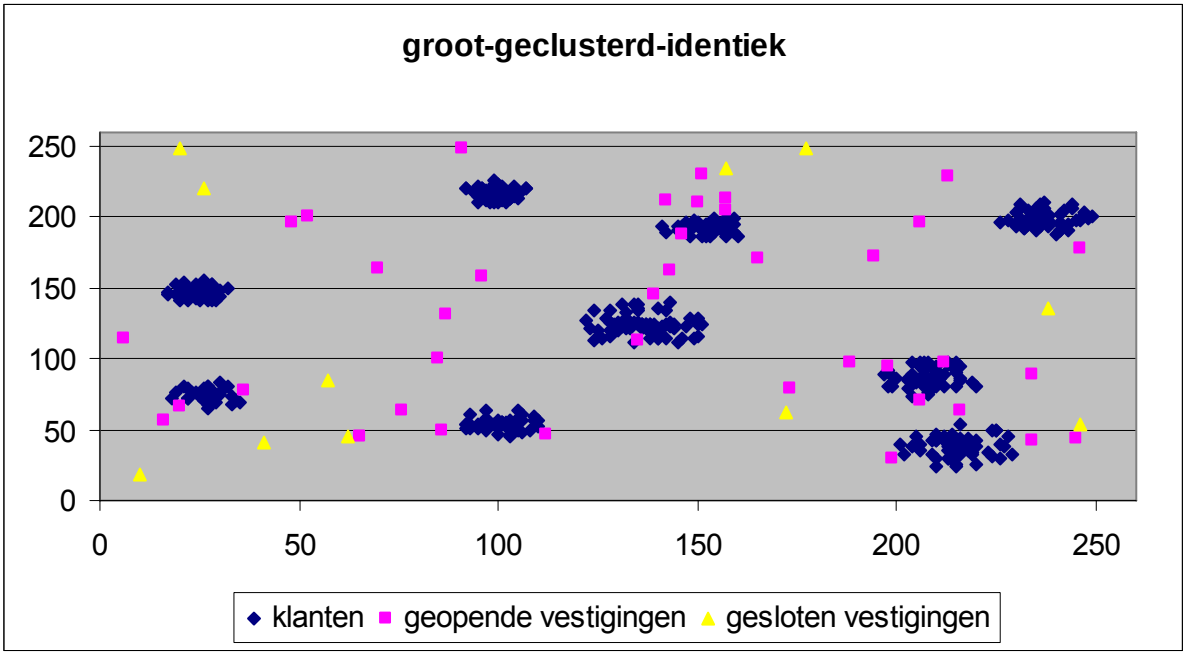
Middel-willekeurig-identiek

	X	Y	capaciteit	kost		X	Y	capaciteit	kost
1	26	26	50	5000	21	113	28	50	5000
2	94	85	50	5000	22	112	69	50	5000
3	116	124	50	5000	23	112	7	50	5000
4	58	34	50	5000	24	41	55	50	5000
5	4	106	50	5000	25	123	13	50	5000
6	62	28	50	5000	26	49	93	50	5000
7	123	137	50	5000	27	57	103	50	5000
8	25	26	50	5000	28	60	131	50	5000
9	86	41	50	5000	29	128	126	50	5000
10	42	4	50	5000	30	131	134	50	5000
11	73	86	50	5000	31	44	4	50	5000
12	19	71	50	5000	32	98	9	50	5000
13	47	37	50	5000	33	57	136	50	5000
14	10	61	50	5000	34	100	118	50	5000
15	52	68	50	5000	35	41	104	50	5000
16	104	139	50	5000	36	14	126	50	5000
17	66	42	50	5000	37	53	68	50	5000
18	97	32	50	5000	38	135	116	50	5000
19	131	59	50	5000	39	114	128	50	5000
20	24	83	50	5000	40	1	48	50	5000



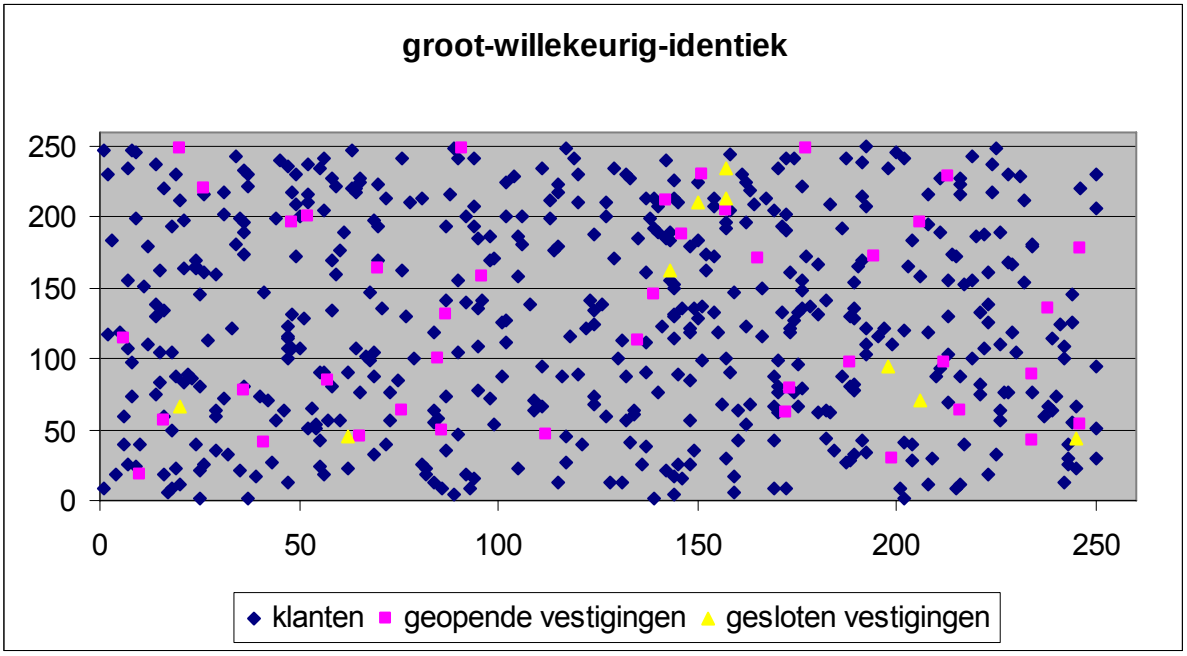
Groot-geclusterd-identiek

	X	Y	capaciteit	kost		X	Y	capaciteit	kost
1	146	188	110	11000	26	216	63	110	11000
2	85	100	110	11000	27	91	249	110	11000
3	87	132	110	11000	28	139	145	110	11000
4	65	45	110	11000	29	10	19	110	11000
5	20	249	110	11000	30	234	89	110	11000
6	62	45	110	11000	31	234	42	110	11000
7	194	173	110	11000	32	143	163	110	11000
8	151	231	110	11000	33	157	214	110	11000
9	135	113	110	11000	34	52	200	110	11000
10	41	41	110	11000	35	96	158	110	11000
11	157	235	110	11000	36	177	249	110	11000
12	245	44	110	11000	37	142	212	110	11000
13	86	49	110	11000	38	70	164	110	11000
14	173	79	110	11000	39	212	98	110	11000
15	206	197	110	11000	40	213	229	110	11000
16	198	95	110	11000	41	76	64	110	11000
17	20	66	110	11000	42	48	196	110	11000
18	157	205	110	11000	43	238	135	110	11000
19	246	53	110	11000	44	26	220	110	11000
20	112	46	110	11000	45	150	211	110	11000
21	206	70	110	11000	46	188	97	110	11000
22	246	178	110	11000	47	172	62	110	11000
23	165	171	110	11000	48	36	78	110	11000
24	6	115	110	11000	49	57	85	110	11000
25	199	29	110	11000	50	16	57	110	11000



Groot-willekeurig-identiek

	X	Y	capaciteit	kost		X	Y	capaciteit	kost
1	146	188	110	11000	26	216	63	110	11000
2	85	100	110	11000	27	91	249	110	11000
3	87	132	110	11000	28	139	145	110	11000
4	65	45	110	11000	29	10	19	110	11000
5	20	249	110	11000	30	234	89	110	11000
6	62	45	110	11000	31	234	42	110	11000
7	194	173	110	11000	32	143	163	110	11000
8	151	231	110	11000	33	157	214	110	11000
9	135	113	110	11000	34	52	200	110	11000
10	41	41	110	11000	35	96	158	110	11000
11	157	235	110	11000	36	177	249	110	11000
12	245	44	110	11000	37	142	212	110	11000
13	86	49	110	11000	38	70	164	110	11000
14	173	79	110	11000	39	212	98	110	11000
15	206	197	110	11000	40	213	229	110	11000
16	198	95	110	11000	41	76	64	110	11000
17	20	66	110	11000	42	48	196	110	11000
18	157	205	110	11000	43	238	135	110	11000
19	246	53	110	11000	44	26	220	110	11000
20	112	46	110	11000	45	150	211	110	11000
21	206	70	110	11000	46	188	97	110	11000
22	246	178	110	11000	47	172	62	110	11000
23	165	171	110	11000	48	36	78	110	11000
24	6	115	110	11000	49	57	85	110	11000
25	199	29	110	11000	50	16	57	110	11000



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:
Het gebruik van memetische algoritmen bij operationele beslissingen. Een toepassing op het vestigingsprobleem
Richting: **Handelsingenieur** 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,

Stéphanie JANSEN

Datum: