Evolutionaire technieken voor optimalisatie van bedrijfsprocessen

BWI-werkstuk

Chi Kit Kong
augustus 2003

Evolutionaire technieken voor optimalisatie van bedrijfsprocessen

Chi Kit Kong
BWI-werkstuk

vrije Universiteit
Faculteit der Exacte Wetenschappen
De Boelelaan 1081a
1081 HV Amsterdam

augustus 2003

Samenvatting

Onlangs zijn verschillende technieken gebaseerd op de principes van de natuur met succes toegepast op verschillende complexe problemen in optimalisatie, systeemherkenning, datamining en andere gebieden. De bekendste technieken zijn: “simulated annealing”, neurale netwerken en evolutionaire technieken. Hoewel globale convergentie is bewezen voor deze technieken, moeten ze gezien worden als heuristieken. De snelst groeiende onderzoeksdomein is evolutionaire techniek, omdat deze methode op verschillende gebieden toegepast kan worden.

Mij is gevraagd een onderzoek te verrichten naar de optimalisatie van bedrijfsprocessen met behulp van evolutionaire techniek. De centrale vraag luidt als volgt:

Op welke wijze kan evolutionaire techniek een bijdrage leveren bij de optimalisatie van bedrijfsprocessen?

Evolutionaire technieken zijn krachtige zoek- en optimalisatietechnieken gebaseerd op het mechanisme van natuurlijke evolutie. Deze technieken imiteren, op een abstracte niveau, biologische principes zoals populatie gebaseerde benadering, erving van informatie en de selectie van kandidaat oplossingen gebaseerd op de kwaliteit. De vier belangrijkste componenten van evolutionaire techniek zijn: selectie, recombinatie, mutatie en de stop conditie.

Evolutionaire technieken worden in de praktijk op verschillende gebieden toegepast, zoals bij verspreiding, roosters optimaliseren, projectselectie, routeringsproblemen, budgettering, investeringsanalyse, betalingsregelingen, ontwerpproblemen, procesbeheersing, netwerkontwerp, enzovoort. Dit werkstuk zal een introductie geven van enkele klassieke optimalisatieproblemen. We bespreken de volgende optimalisatieproblemen:
- Het kortste weg probleem;
- Het handelsreizigersprobleem;
- Het transportprobleem;
- Het knapzakprobleem.

Daarna bespreken we het verschil tussen de sterke en de zwakke oplossingsmethoden.

Na het theoretische gedeelte volgt een beschrijving van toepassingen van evolutionaire techniek. Als eerste worden vijf toepassingen besproken zonder details van de evolutionaire algoritme. De vijf toepassingen zijn: recepten voor huisdiervoedsel, ketenintegratie, reclames, routingtabellen en verkeerslichten.

Daarna bespreken we acht voorbeelden met details van de evolutionaire algoritme. De acht voorbeelden zijn:
- Capaciteitsprobleem van vrachtwagens;
- 2 dimensionale “Cutting”probleem;
- Roosters opstellen;
- Personeelsplanning;
- Productieplanning van identieke machines;
- Insteltijden van machines;
- Seriegewijze productie;
- Positioneren van vliegtuigen.

Evolutionaire technieken zijn breed toepasbare zoek- en optimalisatietechnieken, gebaseerd op het mechanisme van natuurlijke evolutie. De techniek heeft zich bewezen als een van de belangrijkste technologie voor het bouwen van adaptieve systemen. Het systeem is instaat om met complexe en veranderde omgevingen om te gaan, net zoals bij de natuurlijke evolutie. Interesse in evolutionaire techniek is in de laatste jaren dramatische toegenomen en de techniek is met succes toegepast op verscheidene complexe problemen.
Concluderend, evolutionaire techniek is een veelbelovende techniek, toepasbaar op vele complexe optimalisatie problemen. Integratie van evolutionaire techniek met probleemspecifieke kennis heeft grote potentie voor praktische applicaties op het gebied van operationele research. Hoewel het ontwikkelen van hybridische evolutionaire techniek systemen meer kennis en middelen vereist, levert het systeem een uitstekende performance op, door de combinatie van verschillende benaderingen in een geïntegreerd systeem.

Voorwoord

Dit werkstuk is gemaakt in het kader van de BWI-werkstuk voor de studie Bedrijfswiskunde en Informatica aan de Vrije Universiteit. Het BWI-werkstuk is een onderdeel van de studie BWI en vormt het resultaat van een literatuuronderzoek en/of een eigen onderzoek. Dit BWI-werkstuk is voornamelijk gericht op literatuuronderzoek.

Ik wil van de gelegenheid gebruik maken om mijn begeleider te bedanken zonder wie het niet mogelijk zal zijn geweest om dit werkstuk te schrijven. Graag wil ik Prof. Dr. A.E. Eiben bedanken voor zijn begeleiding en adviezen. Dankzij zijn goede adviezen is het een werkstuk geworden waar ik met veel plezier aan heb gewerkt.

Chi Kit Kong

Amsterdam, augustus 2003

Inhoudsopgave

Hoofdstuk 1 Inleiding

1.1 Algemeen

Onlangs zijn verschillende technieken gebaseerd op de principes van de natuur met succes toegepast op verschillende complexe problemen in optimalisatie, systeemherkenning, datamining en andere gebieden. De bekendste technieken zijn: “simulated annealing”, neurale netwerken en evolutionaire technieken. Hoewel globale convergentie is bewezen voor deze technieken, moeten ze gezien worden als heuristieken. De snelst groeiende onderzoeksdomein is evolutionaire techniek, omdat deze methode op verschillende gebieden toegepast kan worden.
Evolutionaire technieken zijn krachtige zoek en optimalisatie technieken gebaseerd op het mechanisme van natuurlijke evolutie. Deze technieken imiteren, op een abstract niveau, biologische principes zoals populatie gebaseerde benadering, erving van informatie en de selectie van kandidaat oplossingen gebaseerd op de kwaliteit.
Traditioneel zijn de meeste praktische applicaties van evolutionaire technieken ontwikkeld voor de technische sector. Voor lange tijd zijn de managementproblemen genegeerd als onderzoeksdomein voor evolutionaire technieken. Dit is verrassend, aangezien de geweldige potentie van evolutionaire benadering voor het zakelijke en economische domein al erkend is in de vroegere publicaties. De eerste persoon die over dit onderwerp schreef is John Holland met zijn boek Adaptation in Natural and Artificial Systems (The University of Michigan Press, 1975).
Het afgelopen decennium is evolutionaire techniek een snel groeiend onderzoeksdomein. Tegenwoordig zijn veel interessante artikelen van evolutionaire techniek gerelateerd aan managementproblemen gepubliceerd. Dit werkstuk geeft een overzicht van evolutionaire techniek in de managementapplicaties. We focussen ons op optimalisatie en wel vanuit het standpunt van operationele research. Evolutionaire techniek mag gezien worden als de nieuwe techniek voor het oplossen van operationele research vraagstukken.

1.2 Onderzoeksvragen

Mij is gevraagd een onderzoek te verrichten naar de optimalisatie van bedrijfsprocessen met behulp van evolutionaire techniek. De centrale vraag luidt als volgt:

Op welke wijze kan evolutionaire techniek een bijdrage leveren bij de optimalisatie van bedrijfsprocessen?

Om deze vraag op een goede manier te beantwoorden is het noodzakelijk om eerst enkele subvragen te beantwoorden. De antwoorden van deze vragen vormen samen het antwoord op de onderzoeksvraag. De vragen die in dit werkstuk beantwoord worden, zijn de volgende:

- Wat is evolutionaire techniek volgens de theorie?
- Bij welke bedrijfsprocessen kan evolutionaire techniek gebruikt worden?
- Op welke wijze wordt evolutionaire techniek toegepast voor optimalisatie van bedrijfsprocessen in de praktijk?
- Wanneer is evolutionaire techniek geschikt voor optimalisatie van bedrijfsprocessen?

1.3 Beperkingen

Wegens bedrijfsprivacy kunnen we vaak de evolutionaire algoritmen niet onderzoeken. In de literatuur worden vaak resultaten van een project en de globale toepassing van evolutionaire techniek besproken. Gedetailleerde beschrijvingen komen we in de literatuur meestal niet tegen.

1.4 Verantwoording van de opzet

Het onderzoek omvat de volgende stappen:

- Ten eerste zal er kort worden ingegaan op de theorie van evolutionaire techniek, zodat een beeld wordt verkregen wat evolutionaire techniek inhoudt;
- Hierna wordt beschreven bij welke bedrijfsprocessen we evolutionaire techniek kunnen tegenkomen;
- Na het theoretische gedeelte volgt een beschrijving van toepassingen. In deze beschrijving komen toepassingen aan bod waar evolutionaire techniek met succes is toegepast;
- Bovenstaande stappen zullen leiden tot een conclusie of evolutionaire techniek geschikt is om optimalisatieproblemen in bedrijfsprocessen op te kunnen lossen.

1.5 Methode van onderzoek

Oriëntatie fase
In de oriëntatie fase wordt het onderzoeksdomein verder afgekaderd. Hiervoor zal bestaande literatuur over evolutionaire technieken en toepassing van evolutionaire technieken worden geraadpleegd en bestudeerd.

Analyse van de praktijk
In de analyse van de praktijk worden verschillende toepassingen in kaart gebracht. Er wordt onderzocht in hoeverre evolutionaire techniek met succes is toegepast. Waar mogelijk worden voor verschillende toepassingen details van de evolutionaire algoritmen gegeven.

Conclusie
In de conclusie geven we aan of evolutionaire techniek in de praktijk toepasbaar is voor het oplossen van optimalisatieproblemen in bedrijfsprocessen.

Hoofdstuk 2 Evolutionaire technieken

2.1 Inleiding

Evolutionaire technieken (ET) zijn zoekmethoden die proberen om door het nabootsen van Darwins "survival of the fittest" principe van de natuurlijke evolutie, omstandigheden te scheppen waarin voortgaande verbetering van oplossing voor het gegeven zoekprobleem optreedt [1, 2].
We beginnen in paragraaf 2.2 met een korte beschrijving van de kenmerken van evolutionaire technieken.
Evolutionaire technieken bestaan uit een aantal belangrijke componenten, namelijk selectie, recombinatie, mutatie en de stop conditie. De verschillende componenten worden in paragraaf 2.3 besproken. We besluiten het hoofdstuk in paragraaf 2.4 met twee convergentie theorieën.

2.2 Kenmerken van evolutionaire technieken

Als eerste moeten we opmerken dat evolutionaire techniek een algemene benaming is. Alle methoden die geïnspireerd zijn door het principe van natuurlijke evolutie classificeren we als een evolutionaire techniek. Er zijn 4 bekende modellen in de wereld van evolutionaire optimalisatie: genetische algoritme, evolutionaire programmering, evolutionaire strategieën en genetische programmering. In dit werkstuk zullen we de vier methoden niet uitgebreid gaan bespreken, maar we bespreken de evolutionaire technieken in het algemeen.

Alle evolutionaire technieken hebben de volgende 4 eigenschappen:
1. Ze gebruiken een populatie van mogelijke oplossingen voor het gegeven probleem;
2. Ze hebben een voortplantingsfase waarin nieuwe oplossingen ("kinderen") worden gevormd die eigenschappen van oude oplossingen ("ouders") erven;
3. Ze beoordelen de kwaliteit van de oplossingen met een doelfunctie die afhankelijk is van het zoekprobleem en baseren de selectie van de oplossingen die zich mogen voortplanten op deze kwaliteitsmaat;
4. ET zijn stochastisch.

De algemene operaties van een standaard ET zijn zeer eenvoudig. Het bevat een populatie van kandidaat oplossingen, $P(t) = \{x_1^t, ... ,x_N^t\} \in I^N$, met I een verzameling van kandidaat oplossingen, N de populatie grootte, en t de generatie. Deze kandidaat oplossingen zijn oplossingen van de doelfunctie $F(x_i^t)$, $i \in 1, ..., N$ dat geoptimaliseerd moet worden om het gegeven probleem op te lossen.

De pseudo code van een standaard ET ziet als volgt uit:

INITIALISEREN populatie met willekeurige kandidaat oplossing
BEREKEN KWALITEIT van elke kandidaat oplossing
    while not *STOP-CRITERIUM* do
        SELECTEER ouders
        RECOMBINEER onderdelen van ouders
        MUTEER de kinderen
        BEREKEN KWALITEIT van de nieuwe kandidaat oplossingen
        VERVANG sommige ouders door hun kinderen
    od
Figuur 2.1: Pseudo code van een standaard ET [1]

2.3 Evolutionaire componenten

Evolutionaire techniek bevat vier belangrijke componenten, namelijk:
- Selectie;
- Recombinatie;
- Mutatie;
- Stop conditie.

De vier componenten van ET zullen in de komende subparagrafen behandeld worden. Ter verduidelijking van de verschillende componenten zullen we de kandidaat oplossingen representeren als een bitstring[1]. Onze bitstring ziet er als volgt uit: $\{0,1\}^5$.

[1] Genetische algoritme representeert kandidaat oplossingen als bitstring ($\{0,1\}^L$), met L de lengte van de bitstring.

2.3.1 Selectie

Selectie is het proces van evaluatie van de kandidaat oplossingen en selecteert de kandidaat oplossingen die zich mogen voortplanten. We hebben een doelfunctie nodig om de kwaliteit van de kandidaat oplossingen in een populatie te kunnen bepalen. In paragraaf 2.2 hebben we F gedefinieerd als de doelfunctie en we definiëren de kwaliteit van een kandidaat oplossing x_i^t als $f(x_i^t)$.
De meest simpelste vorm van selectie is de "roulette wheel" selectie. Stel we willen de doelfunctie $F(x) = x^2$ gaan maximaliseren. We gebruiken een populatie van vier bitstrings met de kwaliteit van de kandidaat oplossingen zoals in tabel 2.1.

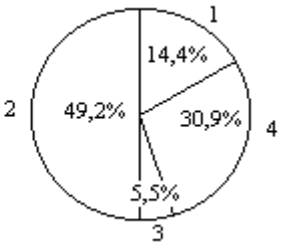
De kans dat een kandidaat oplossing x_i^t wordt geselecteerd voor voortplanting is dan:

$$P(x_i^t) = \frac{f(x_i^t)}{\sum_{i=1}^{n} f(x_i^t)}$$

De "roulette wheel" selectie toegepast op tabel 2.1 is te zien in figuur 2.2.

Nummer	$\{0,1\}^5$	Kwaliteit	% van totaal
1	01101	169	14,4
2	11000	576	49,2
3	01000	64	5,5
4	10011	361	30,9
Totaal		1170	100,0

Tabel 2.1: Voorbeeld van strings met de kwaliteit van de kandidaat oplossingen, van [1]



Figuur 2.2: De "roulette wheel" selectie toegepast op tabel 2.1

Hoe hoger het percentage hoe groter de kans is om geselecteerd te worden om te mogen reproduceren. De "roulette wheel" selectie is één van de bekendste selectiemethoden, andere bekende selectiemethoden zijn "ranking" [3] en "tournament" selectie [4].

2.3.2 Recombinatie

Recombinatie is een component dat uit twee ouders twee kinderen creëert. Met ouders bedoelen we twee kandidaat oplossingen uit de populatie. Kinderen zijn twee nieuwe kandidaat oplossingen, ze ontstaan door uitwisseling van onderdelen van hun ouders.
De meest simpele vorm van recombinatie is een "1- point crossover". Als we het voorbeeld gebruiken, dan gaat recombinatie als volgt: een getal k wordt willekeurig gekozen, met k is kleiner of gelijk aan 5 (lengte van de bitstring). Twee nieuwe kandidaat oplossingen worden gecreëerd door verwisseling van alle karakters tussen positie k+1 en 5. Een voorbeeld van een "1- point crossover" op bitstring 1 en 2 uit tabel 2.1, waar k is 2, is gegeven in figuur 2.3.

$$\left.\begin{array}{l} 1.\ \ 01|101 \\ 2.\ \ 11|010 \end{array}\right\} \Rightarrow \left\{\begin{array}{l} 1'.\ \ 01|010 \\ 2'.\ \ 11|101 \end{array}\right.$$

Figuur 2.3: "1- point crossover" op bitstring 1 en 2 uit tabel 2.1

Een andere bekende recombinatie methode is de "2-points crossover" [5]. De keuze van recombinatiemethoden is afhankelijk van de representatie. Als de representatie van de kandidaat oplossing een binaire boom is dan zal een "2-points crossover" hier minder goed werken.

2.3.3 Mutatie

Mutatie is het willekeurig veranderen van de representatie. Op ieder positie van de bitstring, wordt willekeurig bepaald of op die positie het bit veranderd moet worden. Het veranderen van het bit hangt af van de kans p_m , met $p_m \in [0,1)$. Men noemt p_m wel de mutatieratio. Een voorbeeld van een mutatie op bitstring 1 uit tabel 2.1 is gegeven in figuur 2.4.

1'. 01010 ⇒ 1''. 01111

Figuur 2.4: Dubbele mutatie

Mutatie zorgt voornamelijk voor verscheidenheid in de populatie om te voorkomen dat de populatie vroegtijdig convergeert naar een suboptimum. Evolutionaire techniek zonder mutatie kan leiden tot het verlies van belangrijke informatie. De selectie procedure kan belangrijke informatie elimineren van de populatie die met geen mogelijkheid weer terug te krijgen is. Dit is duidelijk een nadelig gedrag.
Bij hoge mutatieratio zal het algoritme niet convergeren, omdat er teveel nadruk op de verkenning van het gebied ligt. Aan de andere kant, te lage mutatieratio zal leiden tot een vroegtijdige convergentie naar een suboptimum.

2.3.4 Stop conditie

Evolutionaire techniek kent drie stop criteria, namelijk:
1. Men is tevreden met de gevonden oplossing;
2. Het maximaal aantal generatie is bereikt;
3. Alle kandidaat oplossingen zijn identiek.

Het laatste criterium hangt af van de mutatieratio. Een hoge mutatieratio zorgt dat een populatie niet convergeert. Een lage mutatieratio of geen mutatie zal resulteren in een vroegtijdige convergentie, zoals uitgelegd hierboven.

2.4 Convergentie

Als we het gedrag van de evolutionaire technieken analyseren kunnen we twee soorten convergentie onderscheiden, namelijk:
1. Bijna convergentie naar het globale optimum;
2. Populatieconvergentie.

2.4.1 Bijna convergentie naar het globale optimum

Evolutionaire techniek zoekt naar het globale optimum f^* van de doelfunctie F. Stel $Z(t) = \max\{f(x_i^t) \mid x_i^t \in I^N\}$ is de beste kandidaat oplossing van de populatie in generatie t. Een evolutionaire techniek[2] convergeert naar het globale optimum als en slechts dan als

$$\lim_{t\to\infty} P\{Z(t) = f^*\} = 1, \qquad (2.1)$$

waar $P\{Z(t) = f^*\}$ is de kans dat Z(t) is gelijk aan f^*.
In [6] is bewezen dat het standaard genetische algoritme niet convergeert, zoals hierboven staat beschreven. Door mutatie is de kans groter dan nul dat het algoritme het optimum vindt. Het globale optimum wordt gevonden door bij elk generatie de beste oplossing op te slaan. Gebruik van "elitist" selectie, waar de beste kandidaat oplossing van de vorige generatie altijd overleeft, zal zeker leiden tot convergentie zoals beschreven staat in formule (2.1).

"Elitist" selectie
Er is altijd een kans dat de goede kandidaat oplossingen vervangen worden door de minder goede kandidaat oplossingen van de nieuwe generatie, als we de standaard selectiecomponent gebruiken van de evolutionaire techniek. Om dit te voorkomen gebruiken we een "beste" selectie strategie. De strategie houdt in dat de beste kandidaat oplossing of de beste n% van de populatie altijd doorgaat naar de volgende generatie. Een andere mogelijkheid is telkens 1 kandidaat oplossing vervangen uit de populatie. De slechtste kandidaat oplossing wordt het eerst vervangen [6].

[2] Het bewijs voor convergentie naar het globale optimum is gegeven voor genetische algoritme.

2.4.2 Populatieconvergentie

Een ander type convergentie is de populatieconvergentie. Stel $H(x_i^t, x_j^t)$ is de "Hamming" afstand tussen kandidaat oplossingen x_i^t en x_j^t in generatie t, met N de populatiegrootte. Stel

$$\overline{H}(t) = \frac{1}{N^2} \sum_{x_i, x_j \in I^N} H(x_i^t, x_j^t)$$

is de gemiddelde "Hamming" afstand in de populatie in generatie t. De populatie convergeert als en slechts dan als

$$\lim_{t \to \infty} \overline{H}(t) = 0$$

Als de populatie convergeert dan zijn alle kandidaat oplossingen identiek.

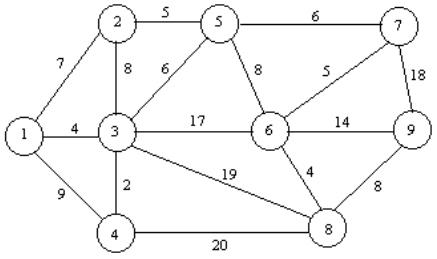
Hoofdstuk 3 Optimalisatieproblemen in de bedrijfsprocessen

3.1 Inleiding

Evolutionaire technieken worden in de praktijk op verschillende gebieden toegepast, zoals bij verspreiding, roosters optimaliseren, projectselectie, routeringsproblemen, budgettering, investeringsanalyse, betalingsregelingen, ontwerpproblemen, procesbeheersing, netwerkontwerp, enzovoort. Wegens de duur van het onderzoek is het niet mogelijk onderzoek te doen naar alle toepassingsgebieden. In dit werkstuk richten we ons voornamelijk op het gebied van operationele research.
In een bedrijf wordt het management vaak met verschillende soorten problemen geconfronteerd, zoals welke projecten moeten we aannemen om de grootste winst te behalen of het optimaal inzetten van het personeel voor een project. In dit hoofdstuk zal een introductie gegeven worden van enkele klassieke optimalisatieproblemen. Hierbij beperken wij tot de meest voorkomende problemen, teneinde door de bomen het bos nog te kunnen blijven zien. Gestart wordt in paragraaf 3.2 met een introductie van het kortste weg probleem. In paragraaf 3.3 komt het handelsreizigersprobleem aan de orde. In paragraaf 3.4 behandelen we het transportprobleem en het toewijzigingsprobleem. Daarna wordt in paragraaf 3.5 de aandacht gericht op het knapzakprobleem. Tenslotte in paragraaf 3.6 bespreken we het verschil tussen de sterke en de zwakke oplossingsmethoden.

3.2 Het kortste weg probleem

Kortste weg probleem komt veelvuldig voor als deelproblemen van grote vraagstukken op het gebied van transport- en communicatienetwerken. Bij de kortste weg probleem proberen we de kortste weg in een bepaald netwerk te vinden [9, 10]. De kortste weg probleem is een van de vele problemen uit de praktijk, die grafisch kunnen worden weergegeven door een situatieschets, zoals in figuur 3.1 gegeven is.
Figuur 3.1: Grafische presentatie van het kortste weg probleem



De cirkels stellen steden (of verkeersknooppunten) voor, terwijl de rechte lijnen de wegverbindingen tussen deze steden zijn. De getallen bij de wegverbindingen stellen de lengtes(afstanden, tijd of kosten) van deze verbindingen voor. U bevindt zich in stad 1 en wilt naar stad 9 via de kortste weg in de bovenstaande schets.

Toepassing van het kortste weg probleem vindt men terug op het gebied van telecommunicatie, computernetwerken, fysieke distributie en projectplanning.

3.2.1 Formulering van het basisprincipe van het kortste weg probleem

Zij
c_{ij} = kosten als verbinding(i, j) doorlopen wordt

$$x_{ij} = \begin{cases} 1 & \text{als verbinding (i, j) doorlopen wordt (in de richting van i naar j)} \\ 0 & \text{anders} \end{cases}$$

Het kortste probleem (weergegeven in figuur 3.1) luidt:

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

onder de voorwaarden:

$$\sum_{j \in V} x_{1j} = 1 \qquad (1)$$

$$\sum_{i \in V} x_{i9} = 1 \qquad (2)$$

$$\sum_{j \in V} x_{ij} - \sum_{j \in V} x_{ji} = 0 \qquad i \in V \qquad (3)$$

$$x_{ij} \in \{0,1\} \qquad i, j \in V \qquad (4)$$

Hierin is V = {1, 2, … , 8, 9} de verzameling knooppunten. De eerste restrictie (1) geeft aan dat je vanuit knooppunt 1 moet vertrekken, de tweede restrictie (2) geeft aan dat je in knooppunt 9 moet arriveren. Voor alle overige knooppunten geldt dat er evenveel verbinding vertrekt als aankomt (3). Tenslotte wordt alleen met stroomwaarden van 0 of 1 gewerkt (4).

3.3 Het handelsreizigersprobleem

Het probleem is van het volgende type aan de orde: een handelsreiziger wil vanuit zijn huis in een nog te kiezen volgorde een aantal klanten precies éénmaal bezoeken en pas op het eind van de rit thuis terugkeren [10]. De afstanden (of kosten of reistijden) tussen zijn huis en de klanten en tussen de klanten onderling zijn bekend. In welke volgorde moet de reiziger de klanten bezoeken als hij de totale afstand (of kosten of reistijd) vanuit zijn rondrit wil minimaliseren?

Het bovenstaande probleem wordt het handelsreizigersprobleem (“Traveling Salesman Problem” ofwel TSP) genoemd. Binnen de fysieke distributie neemt het handelsreizigersprobleem natuurlijk een belangrijke plaats in. Bekende problemen uit dit vakgebied met een identieke structuur zijn namelijk de volgordeproblemen bij het afleveren van goederen, zoals het bezorgen van pakketten van een postorderbedrijf bij particulieren en het bevoorraden met bijvoorbeeld brood bij een keten van supermarkten, als ook bij het vergaren van goederen en het verzamelen van de post uit de brievenbussen van de TPG.

In tabel 3.1 is een handelsreizigersprobleem gegeven met 6 steden en de lengtes van de verbindingen.

Van \ Naar	**1**	**2**	**3**	**4**	**5**	**6**
1	-	10	10	8	11	11
2	10	-	8	5	8	11
3	10	8	-	9	14	16
4	8	5	9	-	10	8
5	11	8	14	10	-	6
6	11	11	16	8	6	-

Tabel 3.1: Het handelsreizigersprobleem met 6 steden

Een handelsreiziger woont in stad 1, moet op een zekere dag in een nog te bepalen volgorde de overige steden 2 t/m 6 precies éénmaal bezoeken en zal pas op het eind van de rit in stad 1 terugkeren. Welke volgorde van bezoeken moet hij in zijn rondrit aanbrengen, als hij de totale afstand van deze rondrit wilt minimaliseren? Een mogelijke rondrit is 1-4-3-2-5-6-1 met lengte 46.

Bij een handelsreizigersprobleem met 6 steden (inclusief de startplaats) kunnen dan in principe (6-1)! = 120 te onderzoeken volgorden optreden. Naarmate het aantal steden meer worden, zullen de te onderzoeken volgorden gigantisch zijn.

3.3.1 Formulering van het basisprincipe van het handelsreizigersprobleem

Zij

$$x_{ij} = \begin{cases} 1 & \text{handelsreiziger reist vanuit stad i naar j} \\ 0 & \text{anders} \end{cases}$$

c_{ij} = kosten als route(i, j) doorlopen wordt ($c_{ij} = c_{ji}$, omdat het probleem symmetrisch is)

Minimaliseer de lengte (kosten van de route):

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Onder de voorwaarden:

$$\sum_{j=1}^{n} x_{ji} + \sum_{j=1}^{n} x_{ij} = 2 \qquad 1 \le i, j \le n \qquad (5)$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \le |S| - 1 \qquad \forall S \subseteq \{1,2,..,n\}, S \neq 0, S \neq \{1,2,..,n\} \qquad (6)$$

$$x_{ij} = 0 \qquad \text{als } j \le i \qquad (7)$$

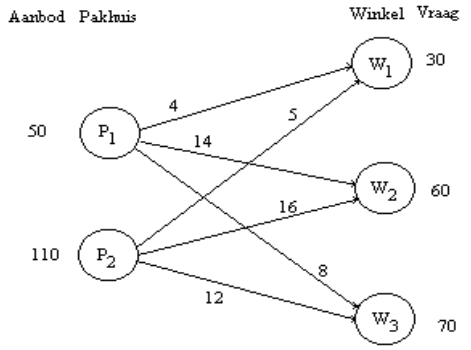
$$x_{ij} = 0 \text{ of } 1 \qquad (8)$$

Alle steden moeten precies 1 keer bezocht worden en alle steden moeten precies 1 keer verlaten worden (5). De restricties (6) zorgen voor subtoereliminatie. We eisen dat $x_{ij} = 0$ voor $j \le i$ (dit voorkomt ook een lus $x_{ii} = 1$). Tenslotte wordt alleen met verbindingen van 0 of 1 gewerkt (4).

3.4 Het transportprobleem

Een zeer bekend probleem op het gebied van planning is het zogeheten transportprobleem [7, 8]. In dit probleem is vaak een situatie van het volgende type aan de orde: maatschappij bezit in verschillende plaatsen fabrieken, waar een zekere hoeveelheid van een bepaalde goed beschikbaar is (of kan worden geproduceerd). Er zijn verscheidene klanten die elk een zekere vraag naar dit goed bezitten. Het wordt vanuit een fabriek rechtstreeks aan een klant geleverd. De vervoerskosten per stuk tussen elke fabriek en elke klant zijn bekend. Het probleem is dan te bepalen welke fabriek hoeveel moet (produceren voor en) leveren aan welke klant als de maatschappij de totale kosten van (productie en) vervoer zo laag mogelijk wil houden en aan de vraag wil houden.

Het transportprobleem in schema vorm is in figuur 3.2 weergegeven.



Figuur 3.2: Het transportprobleem in grafische vorm

De knooppunten P_1 en P_2 stellen twee pakhuizen voor, waar respectievelijk 50 en 110 tonnen meel liggen. De knooppunten W_1, W_2 en W_3 zijn drie winkels, die respectievelijk een vraag van 30, 60 en 70 tonnen meel bezitten. De lijnen stellen verbindingen tussen een pakhuis en een winkel voor. Bij elke lijn staan de kosten van vervoer per ton meel langs die verbinding vermeld. Men wenst het meel uit de pakhuizen te vervoeren naar de winkels, zodanig dat aan de vraag wordt voldaan en de totale kosten van vervoer zo laag mogelijk zijn.

3.4.1 Formulering van het basisprincipe van het transportprobleem

Zij
x_{ij} = de hoeveelheid goederen dat van pakhuis i naar winkel j wordt vervoerd
c_{ij} = de kosten om de goederen van pakhuis i naar winkel j te vervoeren
S_i = aanbod (supply) in pakhuis i, i = 1, .. , m
D_j = vraag (demand) in winkel j, j = 1, .. , n

Het transportprobleem luidt als:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

Onder de voorwaarden:

$$\sum_{j=1}^{n} x_{ij} = S_i \qquad i = 1,..,m \tag{9}$$

$$\sum_{i=1}^{m} x_{ij} = D_j \qquad j = 1,..,n \tag{10}$$

$$x_{ij} \geq 0 \qquad i = 1,..,m \qquad , j = 1,..,n \tag{11}$$

De eerste groep restricties (9) geven aan dat het aanbod van ieder warenhuis wordt verscheept. De tweede groep restricties (10) geven aan dat de vraag bij iedere klant precies voldaan wordt. Tenslotte moeten de te vervoeren goederen niet negatief zijn (11).

3.4.2 Het toewijzigingsprobleem

Een bekende variant van het transportprobleem is het toewijzigingsprobleem [8]. In dit probleem is vaak een situatie van het volgende type: gegeven zijn enige personen en evenveel taken die alleen door deze personen vervuld moeten worden. Het toewijzen van een zekere persoon aan een zekere taak brengt kosten met zich mee. Welke personen moeten welke taak uitvoeren, als men de totale toewijzigingskosten wil minimaliseren en aan elke persoon slechts één taak kan toekennen?
Voorbeelden van het hier optredende probleem kan zijn het optimaal plaatsen van werknemers bij machines, het gunstig toewijzen van aannemers aan projecten en het bij gegeven fabrieksbestand bepalen van de optimale productieplaatsen voor een serie nieuwe producten.

3.5 Het knapzakprobleem

Een knapzakprobleem ziet als volgt uit: een knapzak met een zekere volume dient met een vrij te kiezen deel van een zeker aantal goederen, waarvan het volume en het nut bekend is, zodanig geladen te worden, dat het totale nut van de inhoud van de rugzak zo groot mogelijk is [8]. Het knapzakprobleem komt vaak voor bij een groot aantal verschillende praktische toepassingen zoals projectselectie problemen en investeringsproblemen.
Een iets moderner probleem van dezelfde orde is het volgende: een supermarktketen wil voor maximaal 14 miljoenen euro's een aantal nieuwe vestigingen openen. De keten heeft vier soorten winkels, te weten van type A, B, C en D. Ook de nieuwe te bouwen winkels zullen van type A, B, C en D zijn. De kosten van de vier typen winkels en de (verwachte) omzet in het volgend boekjaar zijn gegeven in tabel 3.2.

Type	Kosten in miljoenen euro's	Omzet in miljoenen euro's
A	4	24
B	3	20
C	2	11
D	1	5

Tabel 3.2: Voorbeeld van een knapzakprobleem

Het management wenst een zo groot mogelijke totale omzet van de nieuwe vestigingen in het komende jaar. Hoeveel winkels van welke type dienen gebouwd te worden?

3.5.1 Formulering van het basisprincipe van het knapzakprobleem

Zij
x_i = het aantal te bouwen winkels, i = 1, .. , n
w_i = omzet van winkel x_i
a_i = kosten om winkel x_i te bouwen
b = maximale bedrag dat gebruikt kan worden voor het bouwen van de winkels

Het knapzakprobleem luidt als:

$$\max \sum_{i=1}^{n} w_i x_i$$

Onder de voorwaarden:

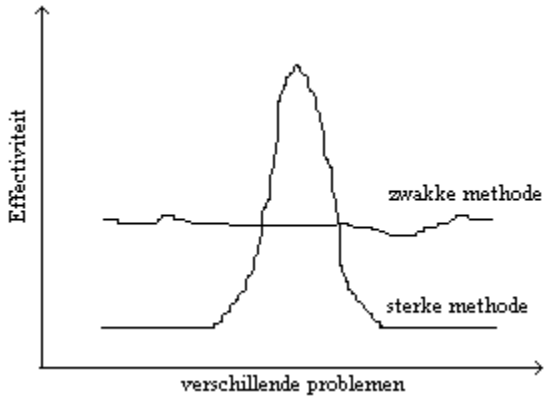
$$\sum_{i=1}^{n} a_i x_i \leq b \qquad (12)$$

$$x_i \geq 0 \qquad i = 1,..,n \qquad (13)$$

De restrictie (12) zorgt voor dat het maximaal te besteden bedrag niet overschreden wordt. Tenslotte moeten de te bouwen winkels niet negatief zijn (13).

3.6 Oplossingsmethoden

Voor het oplossen van bovenstaande problemen (kortste weg probleem, handelsreizigersprobleem, transportprobleem en knapzakprobleem) kunnen we twee groepen oplossingsmethoden onderscheiden: zwakke methode en sterke methode. Een zwakke methode heeft weinig tot geen probleemspecifieke kennis nodig, maar bij de sterke methode hangt het af van deze kennis. Sterke methoden zijn, door het gebruik van probleemspecifieke kennis, beter dan de zwakke methode in het oplossen van het probleem waarvoor ze zijn ontworpen. Vaak kunnen de sterke methoden de optimale oplossingen vinden met weinig rekentijd. Het grootste nadeel is hun kleine toepasbaarheid. We kunnen denken aan de Hongaarse methode [11] voor het toewijzingsprobleem.
Zwakke methoden zijn niet zo efficiënt als de sterke methoden in het oplossen van specifieke problemen, maar ze zijn wel breed toepasbaar. Het verschil tussen de sterke en zwakke methode is geïllustreerd in figuur 3.3



Figuur 3.3: Vergelijking tussen sterke en zwakke methode

Evolutionaire techniek behoort tot de zwakke methode.

3.6.1 Hybridische evolutionaire technieken

In de praktijk komt het voor dat evolutionaire techniek probleemspecifieke kennis gebruikt voor het oplossen van problemen. Deze methode wordt de hybridische evolutionaire techniek genoemd en is een combinatie van de sterke en de zwakke methode. Hybridische evolutionaire techniek bevat zowel de goede eigenschappen van de sterke methode (optimale oplossing vinden in korte tijd) als die van de zwakke methode (vrij snel ontdekken van goede gebieden in een zoekruimte).
In figuur 3.4 zijn de mogelijke plaatsen opgenomen waar probleemspecifieke kennis met de kandidaat oplossing kan fuseren.

INITIALISEREN populatie met willekeurige kandidaat oplossing ←**gebruik probleemspecifieke kennis**
BEREKEN KWALITEIT van elke kandidaat oplossing
    while not *STOP-CRITERIUM* do
        SELECTEER ouders
        RECOMBINEER onderdelen van ouders ←**gebruik probleemspecifieke kennis**
        MUTEER de kinderen ←**gebruik probleemspecifieke kennis**
        BEREKEN KWALITEIT van de nieuwe kandidaat oplossingen
        VERVANG sommige ouders door hun kinderen
    od

Figuur 3.4: Mogelijke plaatsen in een standaard evolutionaire techniek om de kandidaat oplossing te laten fuseren met probleemspecifieke kennis

In het volgende hoofdstuk onderzoeken we hoe deze zwakke methode en de hybridische methode in de praktijk wordt toegepast.

Hoofdstuk 4 Optimalisatie van bedrijfsprocessen door middel van evolutionaire technieken

4.1 Inleiding

In hoofdstuk 3 hebben we enkele klassieke optimalisatieproblemen besproken. In dit hoofdstuk presenteren we ET-applicaties van operationele problemen. De meeste problemen kunnen we indelen in de problemen die we in hoofdstuk 3 hebben besproken of een variant hiervan.
Het hoofdstuk bestaat uit twee delen. In het eerste deel presenteren we een vijftal ET-applicaties zonder de details van het evolutionaire algoritme. De evolutionaire techniek is gebruikt voor het oplossen van het optimalisatieprobleem, maar het gebruikte evolutionaire algoritme is niet vrijgegeven. De vijf voorbeelden zonder details van de evolutionaire algoritmen zijn interessante praktijksituaties. Het laat zien hoe evolutionaire techniek in de praktijk wordt toegepast. Bij het tweede gedeelte geven we acht voorbeelden waar evolutionaire techniek toegepast kan worden. Het verschil met gedeelte één is dat we bij dit gedeelte wel details van het evolutionaire algoritme bij elk voorbeeld opnemen. Enkele, gepresenteerde evolutionaire algoritmen zijn geïmplementeerd in commerciële software. Terwijl andere voorbeelden meer een indicatie geven hoe bepaalde praktijkproblemen opgelost kunnen worden met evolutionaire techniek. De gepresenteerde algoritmen zijn getest met behulp van willekeurig gegenereerde data of gegeven testproblemen uit de literatuur of werkelijke data uit de praktijk.

4.2 Praktijk voorbeelden zonder details van de evolutionaire algoritmen

De vijf voorbeelden die we in deze paragraaf bespreken zijn:
- Recepten voor huisdiervoedsel;
- Ketenintegratie;
- Reclames;
- Routing tabellen;
- Verkeerslichten.

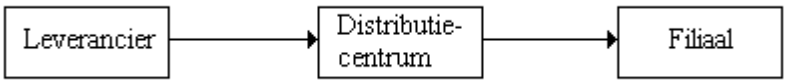
Het voorbeeld reclames is een variant van de klassieke knapzakprobleem. Terwijl het voorbeeld routing tabellen een kortste weg probleem is. De andere voorbeelden zijn optimalisatie problemen die niet tot de klassieke optimalisatie problemen behoren die we in hoofdstuk 3 hebben besproken.

4.2.1 Recepten voor huisdiervoedsel

De belangrijkste peiler voor een goede gezondheid van een hond en kat, is de voeding. Goede voeding is nodig voor het lichaam om zichzelf in stand te houden en om af te rekenen met alle slechte invloeden van buitenaf. Als het lichaam overbelast wordt met afvalstoffen van slechte voeding, kan het niet goed functioneren en zullen de opgestapelde afvalstoffen op den duur ziekten veroorzaken. Dus voedsel voor huisdieren moet van hoge kwaliteit zijn. Het meten van de kwaliteit van het voedsel is zeer subjectief (smaak, aroma, enzovoort). Tijdens het ontwikkelen van recepten moeten bedrijven rekening houden met wetten en voorschriften, maar ze moeten ook rekening houden met producteisen. Het bevredigen van zowel wettelijke als product vereisten lukt vaak niet. Dikwijls kunnen deze vereisten in conflict komen.
Verschillende variaties betekenen dat een reeks van recept ontwerpen nodig zijn om alle vereisten te ontmoeten en dikwijls kunnen deze vereisten in conflict komen. Verschillende vereisten, zoals verschillende ingrediënten, beschikbaarheid van ingrediënten, verandering in bewerkingscondities en lokale product voorkeuren moet allemaal rekening mee worden gehouden. Ontwerpers moeten uit meer dan 200 ingrediënten opties kiezen om een recept te ontwerpen, tegen zo laag mogelijke kosten en toch voldoen aan de gestelde kwaliteit.
Met over 200 ingrediënt opties plus bewerkingsparameters zijn er verschillende bijna optimale oplossingen en veel sub optimale oplossingen. Evolutionaire techniek is gebruikt om dit probleem op te lossen.
De kwaliteit van een recept is afhankelijk van fuzzy regels en van de kosten. De regels zijn van de vorm: als kooktijd groter dan t is dan is de smaak slecht. Het kan zijn dat de kwaliteit van een recept heel goed is maar als de kosten te hoog zijn dan is het nog geen goed recept.
Tijdens het optimalisatieproces kunnen de gebruikers op ieder ogenblik het proces tijdelijk stoppen en zo verschillende wijzigingen aanbrengen. Met een standaard pc duurt de optimalisatie niet meer dan een paar minuten. Voor meer informatie zie [12].

Resultaat
Het project is geslaagd als er recepten zijn gevonden waarbij de kosten met 2% zijn afgenomen. Door het toepassen van evolutionaire techniek zijn hoge kwaliteitsrecepten ontworpen en de kosten zijn met 5% afgenomen.

4.2.2 Ketenintegratie
Ketenintegratie is in feite een logische stap binnen de ontwikkeling van de logistiek. Wil een onderneming in een steeds sterker concurrerende omgeving blijven functioneren, dan zal samenwerking en afstemming zich niet alleen tot interne processen moeten beperken maar zich moeten uitstrekken over de gehele logistieke keten. Ketenintegratie kan zodoende de positie van de keten versterken ten opzichte van de concurrerende keten en daarmee de positie van de ketendeelnemers. Een voorbeeld van een keten is opgenomen in figuur 4.1.



Figuur 4.1: Voorbeeld van een keten

Unilever en A. Ahlstrom weten dat ketenintegratie belangrijk is en daarom hebben ze samen een ketenintegratie optimalisatiesysteem ontwikkeld om de keten te optimaliseren. Unilever is een van de grootste producenten ter wereld van voedingsmiddelen en producten voor huishoudelijke en persoonlijke verzorging. A. Ahlstrom is een leverancier van geavanceerde kunstvezels materialen.
Een belangrijke reden om ketenintegratie in te voeren is het verlagen van kosten en in overleg met aangrenzende schakels binnen de keten. De tweede reden voor ketenintegratie is het verhogen van de customer service. Uniliver en A. Ahlstrom denken dat ze de efficiency van de keten kan verhogen door een ketenintegratie optimalisatiesysteem.
Het systeem optimaliseert het netwerk van leveranciers, transport, productie, magazijnen en klanten. In het systeem is een evolutionaire techniek geïmplementeerd. De techniek kan verbindingen in een netwerk van ketens beschrijven en de logistieke verbindingen ertussen. Het gebruik van een groot aantal alternatieve oplossingen stimuleert het systeem om continue de efficiency van de keten te verhogen. De evolutionaire techniek in het systeem wordt gebruikt om te identificeren en te optimaliseren van de verschillende interne verbindingen van een keten om zo de efficiency te verbeteren. Voor meer informatie zie [13].

Resultaat
Het systeem vermindert zowel de leveringstijden als de logistieke kosten in de keten. De logistieke kosten kunnen verlaagd worden tussen de 5 – 15% en de leveringstijden zijn verminderd met meer dan 30%. Het systeem ondersteunt ook de manager bij het maken van beslissingen.

4.2.3 Reclames
Eén van de inkomsten van een televisie station is het uitzenden van reclames. Men wil zoveel mogelijk reclames uitzenden in een beperkte tijd. Een groot commercieel TV station in de UK had de behoefte om de volgorde van het uitzenden van de reclames binnen een commerciële break te optimaliseren. Een break is de tijd dat de reclame wordt uitgezonden. De zendtijd is beperkt tot 4 minuten en ze moeten ook rekening houden met 6 verschillende regio's die ze gaan uitzenden. In 4 minuten zendtijd over 6 regio's gaat het vaak om 50 reclames. Vaak hebben adverteerders een speciale voorkeur in gedachten zoals aan het begin of aan het eind van een break. Ook wordt vaak gekozen voor een 'kop en staart' in een break, dit houdt in dat een reclame 2 keer wordt uitgezonden in dezelfde break. In dezelfde break mogen advertenties elkaar niet overlappen en er mogen tussen de advertenties geen gaten zitten.
Het probleem kan analytisch opgelost worden. Maar als het aantal combinaties groter wordt is het onpraktisch alle combinaties na te gaan. Een bezwaar is dat men niet binnen een redelijk tijd aan een oplossing kan komen.
Een manier om het bovenstaande probleem op te lossen is met behulp van evolutionaire techniek. Om de lange evolutie tijd te verkorten hebben ze een speciale, heuristische methode gebruikt. De kandidaat oplossingen worden na recombinatie en mutatie gecorrigeerd door de heuristische methode. Het voordeel van deze benadering is dat je altijd een toegelaten kandidaat oplossing krijgt, want de heuristiek houdt rekening met de harde voorwaarden. Deze benadering vermindert de evolutie tijd in seconden. Voor meer informatie zie [14].

Resultaat
Het toepassen van ET voor het bovenstaande probleem levert een redelijke oplossing op in een tamelijk korte tijd, terwijl het vinden van de optimale oplossing misschien oneindige tijd vereist.
In plaats van alleen het eenvoudig beheren van de breaks, kunnen ze nu de zendtijd beter benutten. De opbrengst voor het toepassen van evolutionaire techniek is helaas niet gepubliceerd.

4.2.4 Routingtabellen
Moderne telecommunicatienetwerken bestaan uit twee belangrijke componenten:
- Hardware, inclusief kabels en routers;
- Software, inclusief het algoritme voor de sturing van het gesprek (routingtabellen)

In complexe communicatienetwerken bestaat meer dan 90% van de kosten van het netwerk uit hardware en de installatie van de hardware. Voor een uitbreiding aan het bestaande netwerk voor verkeersverandering moeten nieuwe kabels en switches geïnstalleerd worden.
In de markt van telecommunicatie is de concurrent zeer hoog. Om te overleven richten telecom bedrijven zich op de potentie van nieuwe technologieën om zich staande te houden. Ze richten vooral op technologieën die zich aanpassen aan de omgeving en uit zichzelf leren. Siemens AG, een toonaangevend bedrijf op gebied van de elektrotechniek en elektronica, gelooft dat deze technieken de routingtabellen van gesprekken kunnen verbeteren. Routingtabellen zijn tabellen waarin is opgenomen, de weg die de data moet afleggen om van de ene plaats naar een andere plaats te komen in een netwerk. Gebruik maken van de nieuwe technologieën zal de installatiekosten van de hardware verminderen. De nieuwe technologie zal er voor moeten zorgen dat de huidige netwerkcapaciteit volledig wordt benut. De optimalisatie van de routingtabellen zal de klanttevredenheid verhogen, omdat de kwaliteit van de verbinding hoger en beter is. Siemens realiseert dat dit op lange termijn een concurrentie voordeel oplevert.
AG Siemens ontwikkelt een systeem die de routingtabellen kan optimaliseren. Hiervoor gebruiken ze een evolutionaire benadering. In het nieuwe systeem kan een routingtabel zich aanpassen aan oneindig veel verschillende verkeerssituaties. Voor meer informatie zie [15].

Resultaat
Het nieuwe systeem verbetert de performance met bijna 1000% vergeleken met het oude systeem. Het systeem bepaald het volgende knooppunt die bezocht moet worden op weg naar zijn bestemming, gebaseerd op de huidige situatie.
Het gebruik van het optimalisatiesysteem voor routingtabellen levert voor Siemens AG de volgende voordelen op:
- Een besparing op de netwerkinstallatiekosten;
- Het netwerkontwerpproces is eenvoudig, dit resulteert in de vermindering van de ontwikkelingskosten en de ontwikkelingstijd;
- Siemens kan sneller reageren op de markt.

4.2.5 Verkeerslichten
Het wegennet van Nederland is vol en het land krijgt te maken met toename van verkeer op de wegen. Hoewel de wegen al op maximale capaciteit zitten. Het verkeersprobleem wordt veroorzaakt door kruispunten in de wegen met verkeerslichten. Tijdens files vormen de kruispunten de “bottlenecks”. Dus optimalisatie van deze kruispunten is een belangrijke verbetering over het hele wegennet.
Het Nederlandse Ministerie van Verkeer en Waterstaat wil een systeem hebben dat zich aanpast aan de veranderende verkeerssituaties. Dit houdt in dat de tijdsduur van het groene verkeerslicht varieert, afhankelijk van de huidige verkeerssituatie.
Evolutionaire techniek is gebruikt om dit probleem op te lossen, omdat deze techniek zich gemakkelijk aanpast aan de omgeving. Voor meer informatie zie [16].

Resultaat
Voor het testen van het systeem is gebruik gemaakt van het simulatieprogramma FLAS / FLEXSYT-II. Het programma kan verschillende werkelijke verkeerssituaties simuleren. Vier kruispunten zijn gebruikt voor het testen van het systeem. Het systeem krijgt informatie van de omgeving en deze informatie wordt gebruikt voor de minimalisatie van het aantal stops per voertuig.
Zonder enige aanpassing van de parameters of van de doelfunctie geeft het systeem ongeveer hetzelfde resultaat als de huidige verkeerscontrollers die bij het Nederlandse Ministerie van Verkeer en Waterstaat is gebruikt. Het verschil zit in de aanpasbaarheid van de techniek. Bij onverwachte gebeurtenissen (bijvoorbeeld een ongeluk) past het systeem zich

onmiddellijk aan, aan de omgeving. Het systeem optimaliseert de tijdsduur van het groene verkeerslicht op ieder kruispunt. Afstemming van de parameters en de doelfunctie voor elk kruispunt geeft bij alle scenario's een beter resultaat dan de huidige verkeerscontrollers.
Het Nederlandse Ministerie van Verkeer en Waterstaat heeft het systeem in gebruik genomen.

4.3 Praktijk voorbeelden met details van de evolutionaire algoritmen

De acht voorbeelden die we in deze paragraaf bespreken zijn:
- Capaciteitsprobleem van vrachtwagens;
- 2 dimensionale "Cutting"probleem;
- Roosters opstellen;
- Personeelsplanning;
- Productieplanning van identieke machines;
- Insteltijden van machines;
- Seriegewijze productie;
- Positioneren van vliegtuigen.

Het capaciteitsprobleem van vrachtwagens is een variant van het knapzakprobleem. Roosters opstellen, personeelsplanning, productieplanning van identieke machines en seriegewijze productie zijn allemaal een variant van het toewijziginsprobleem. Insteltijden van machines is een handelsreizigersprobleem. Alleen het 2 dimensionale "cutting" probleem en het positioneren van vliegtuigen zijn optimalisatieproblemen die niet tot de klassieke optimalisatieproblemen behoren die we in hoofdstuk 3 hebben besproken.

4.3.1 Capaciteitsprobleem van vrachtwagens

Dit is een probleem die veel voorkomt in verschillende praktijksituaties. We hebben een aantal items en die moeten vervoerd worden met vrachtwagens. Iedere vrachtwagen heeft een maximumcapaciteit en we hebben maar een aantal vrachtwagens beschikbaar. De bedoeling is om zo min mogelijk vrachtwagens te gebruiken om alle items te vervoeren.

4.3.1.1 Evolutionaire algoritme

Deze strategie is afkomstig van [17]. De strategie werkt met groepen items tegelijk en de recombinatie- en mutatieoperatie werken met groepsonderdelen in de kandidaat oplossing. Om het zoekproces te verbeteren is een speciale vervangingsoperatie toegevoegd. Deze speciale operatie verbetert de kwaliteit van de kandidaat oplossing.

4.3.1.2 Kandidaat oplossing

Stel dat we een aantal items hebben en ieder item geven we een apart nummer. We hebben ook vrachtwagens en ieder vrachtwagen geven we een unieke letter. Een voorbeeld van een kandidaat oplossing kan er als volgt uitzien

0	1	2	3	4	5
A	D	B	C	E	B

Item 0 wordt in vrachtwagen A geladen, 1 in vrachtwagen D, 2 in vrachtwagen B, 3 in vrachtwagen C, 4 in vrachtwagen E en 5 in vrachtwagen B. De kandidaat oplossing kan ook geschreven worden als

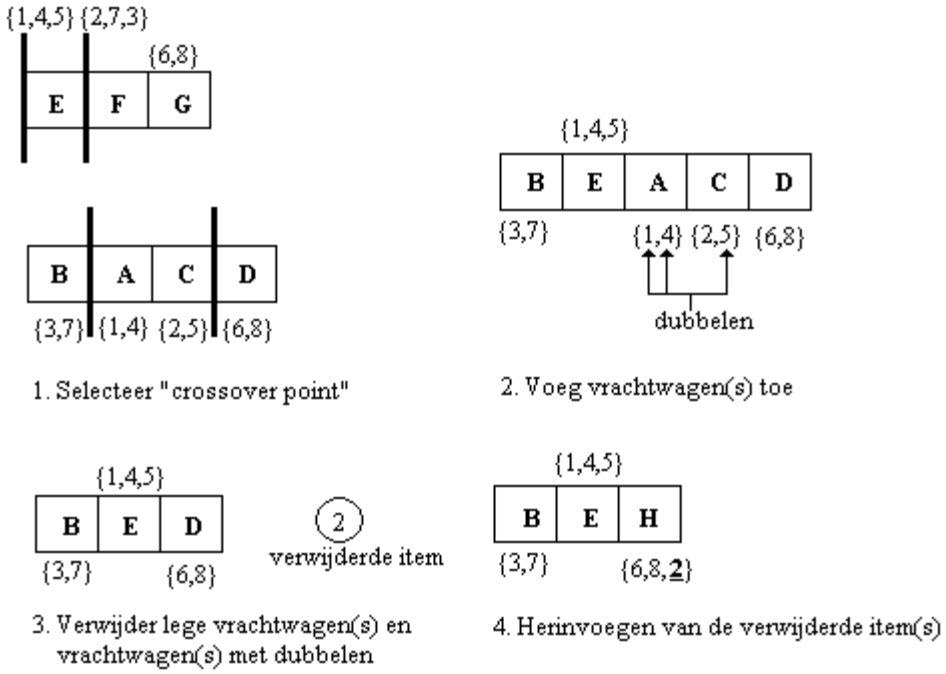
{0}	{2,5}	{3}	{1}	{4}
A	B	C	D	E

In deze representatie werken we met groepen. Bij de verschillende operaties gaan we uit van deze representatie.

4.3.1.3 Recombinatie

De recombinatieoperatie is geïllustreerd in figuur 4.2. De stappen van recombinatie zijn als volgt:
1. Selecteer willekeurig 2 kandidaat oplossingen, bepaal de "2 crossover points" bij beide kandidaat oplossingen.
2. Voeg de inhoud van de "2 crossover points" van de eerste kandidaat oplossing bij de eerste "crossover point" van de tweede kandidaat oplossing.
3. Verwijder items die twee keer voorkomen bij de tweede kandidaat oplossing. Tijdens het verwijderen worden hele groepen verwijderd. Items die 1 keer voorkomen maar toch verwijderd zijn, worden opnieuw aan de kandidaat oplossing toegevoegd.

4. Als het nodig is kunnen we hier de speciale operatie toepassen.
5. Herhaal stap 2 tot en met 4 met dezelfde kandidaat oplossingen, alleen nu met de rollen omgedraaid.



Figuur 4.2: De recombinatieoperatie

4.3.1.4 Kwaliteit van de kandidaat oplossing

De kwaliteit van de kandidaat oplossing wordt bepaald door de volgende doelfunctie:

$$f(x_i) = \frac{\sum_{j=1}^{N} (F_j / C)^2}{N}$$

met x_i is de i-de kandidaat oplossing,
N is het aantal gebruikte vrachtwagens in de kandidaat oplossing,
F_j is het totaal gewicht van de items in vrachtwagen j,
C is de capaciteit van de vrachtwagen.

4.3.1.5 Mutatie

De mutatieoperatie verwijdert willekeurige vrachtwagens. De items in deze vrachtwagens worden opnieuw toegevoegd aan de kandidaat oplossing met de speciale operatie (zie paragraaf speciale operatie).

4.3.1.6 Speciale operatie

In stap 3 worden de verwijderde items weer toegevoegd aan de kandidaat oplossing. Eerst wordt het rijtje items gesorteerd op gewicht, aflopend. De items worden daarna één voor één toegevoegd aan de eerste vrachtwagen met voldoende ruimte. We noemen dit de standaard toevoeging.
De speciale operatie heeft betrekking op het toevoegen van verwijderde items aan een kandidaat oplossing. De operatie controleert de vrachtwagens in de kandidaat oplossing één voor één. De operatie controleert of het mogelijk is om maximaal drie items in een vrachtwagen te vervangen door 1 of 2 items van de verwijderde items. Een eis is dat het totale gewicht van de items in de vrachtwagen moet toenemen zonder de capaciteit van de vrachtwagen te overschrijden. Als het mogelijk is dan wordt de vervanging uitgevoerd. Dit betekent dat items die net niet toegewezen zijn, nu zijn toegewezen. Terwijl sommige items die net toegewezen zijn, nu niet meer zijn toegewezen. Deze operatie wordt herhaald tot dat er geen vervanging meer mogelijk is. Items die overblijven worden weer toegevoegd met de standaard toevoeging.

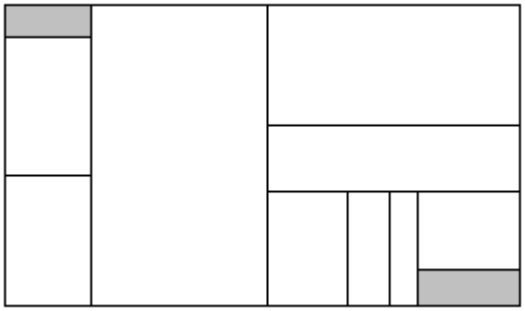
De speciale operatie zorgt er voor dat de vrachtwagens beter gevuld zijn. Eerst worden de grote items toegewezen aan een vrachtwagen en daarna worden de lege ruimtes opgevuld met de kleinere items.

4.3.1.7 Resultaat

Het algoritme is getest met testproblemen uit de literatuur. Het resultaat is vergeleken met een variant van branch-and-bound zoekmethode (MTP). Het algoritme geeft bij 500 tot 1000 items betere kandidaat oplossingen en het zoekproces gaat sneller dan de MTP. Naarmate het aantal items stijgt, geeft het algoritme aanzienlijk betere kandidaat oplossingen dan MTP. Bij moeilijke problemen lukt het algoritme om de optimale oplossing te vinden, terwijl het bij MTP niet lukt.

4.3.2 "Cutting" probleem

In de metaalindustrie komt het vaak voor dat ze uit een plaat verschillende figuren moeten snijden. De bedoeling is de plaat zo te snijden dat het afval (kleine stukjes die overblijven na het snijden) zo min mogelijk is. Optimaal is het als we na het snijden geen afval overhouden. Hieronder presenteren we een algoritme om dit probleem op te kunnen lossen. Het doel is om uit een grote rechthoekige plaat van lengte L en breedte W een set van p kleine stukken te snijden. Elk i-de stuk heeft een oppervlakte van (l_i, w_i). In figuur 4.3 is een toegelaten kandidaat oplossing opgenomen.



Figuur 4.3: Voorbeeld van een toegelaten kandidaat oplossing

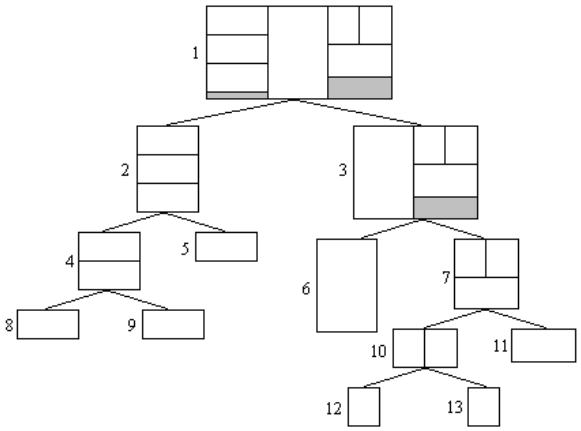
De grijze rechthoeken zijn afval. Met ons probleem kan de machine alleen een horizontale of een verticale snee maken.

4.3.2.1 Evolutionaire algoritme

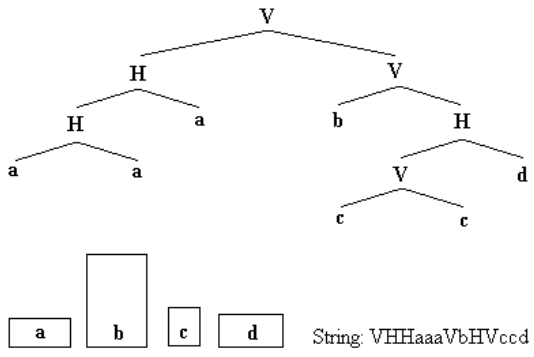
Het algoritme is afkomstig van [17]. Voor de kandidaat oplossing gebruiken we een string representatie. De string representatie is opgebouwd uit de snij operatie en de eindstukken. Het algoritme gebruikt alleen recombinatie met probleemspecifieke kennis. Een kenmerk van het algoritme is het bijhouden van een verzameling van beste kandidaat oplossingen. De verzameling wordt op iedere generatie geüpdate.

4.3.2.2 Kandidaat oplossing

De structuur van onze kandidaat oplossing is een string. Voor de vertaling van de string hebben we een binaire boom nodig. In figuur 4.4 is een voorbeeld opgenomen van een binaire boom met het snijpatroon.

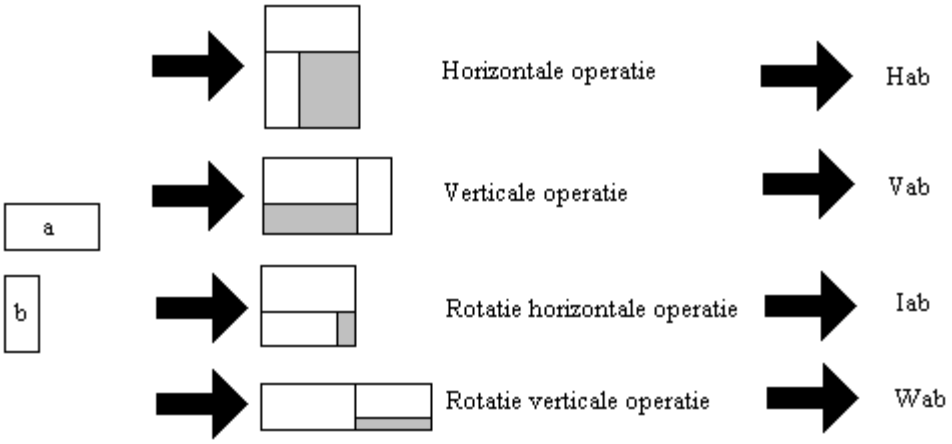


Figuur 4.4: Binaire boom met snijpatroon



Figuur 4.5: Syntactische binaire boom

Elk knop in de boom correspondeert met een verticale of een horizontale snee. Om in figuur 4.4 van 1 naar 2 en 3 te komen moeten we een verticale snee maken. De operaties die uitgevoerd zijn definiëren we met 'V' als verticale snee en 'H' als horizontale snee. Dit doen we ook voor de eindstukken en ieder eindstuk geven we een aparte letter. Gelijke eindstukken krijgen dezelfde letter. Door gebruik te maken van deze notatie kan de binaire boom van figuur 4.4 gepresenteerd worden als een syntactische binaire boom, zie figuur 4.5. Om de string representatie te krijgen moeten we de syntactische boom "depth-first search" doorlopen. Kenmerk van "depth-first search" is dat we bij een knop altijd de linkertak eerst doorlopen en daarna pas de rechtertak. De string representatie VHHaaaVbHVccd van figuur 4.5 correspondeert met het snijpatroon van figuur 4.4.
Om het afval te verminderen is een rotatie patroon van 90° ingevoerd. We definiëren 'W' als de verticale rotatie en 'I' als de horizontale rotatie. In figuur 4.6 zijn twee nieuwe operatoren geïllustreerd.



Figuur 4.6: Twee nieuwe operatoren

4.3.2.3 Recombinatie

De recombinatieoperatie is een concatenatie van twee strings en deze zet aan het begin van de nieuwe string een corresponderende operator. Concatenatie houdt in dat een string achter een andere string wordt geplakt. Als we een string s_i en s_j hebben dan krijgen we de volgende nieuwe strings: 'Vs_is_j', 'Ws_is_j', 'Hs_is_j' en 'Is_is_j'.

4.3.2.4 Kwaliteit van een kandidaat oplossing

De kwaliteit van een kandidaat oplossing hangt af van de ingenomen oppervlakte. De doelfunctie is $F(s_i) = \alpha$ / W*L, met s_i is de i-de kandidaat oplossing, α is de ingenomen oppervlakte van alle stukken, W en L is breedte en lengte van de plaat. Hoe groter $F(s_i)$ hoe beter de kwaliteit van de kandidaat oplossing is.

4.3.2.5 Resultaat

Het algoritme is getest met testproblemen uit de literatuur. Het algoritme is getest met en zonder de rotatieoperator. Uit het resultaat blijkt dat het algoritme met rotatieoperator een stuk beter werkt dan zonder de rotatieoperator. Het algoritme is vergeleken met de MWA[3] methode. Het algoritme zonder rotatieoperator geeft ongeveer dezelfde resultaten als MWA, maar het algoritme met rotatieoperator geeft een betere oplossing dan MWA. Verdere resultaten geven aan dat het algoritme bij probleem gevallen beter en sneller oplossingen genereert dan MWA.

4.3.3 Roosters opstellen

Roosters opstellen gaat om een planning van een aantal ontmoetingen (bijvoorbeeld examens, lessen) waarbij een groep mensen betrokken worden (bijvoorbeeld studenten, leraren) voor een gegeven periode en vereist de benodigde ruimtes (bijvoorbeeld kamers, lokalen, laboratoria). Een dergelijk praktijkprobleem komt vaak voor bij scholen. In de praktijk is het heel moeilijk om roosterproblemen op te lossen, omdat we rekening moeten houden met veel voorwaarden.

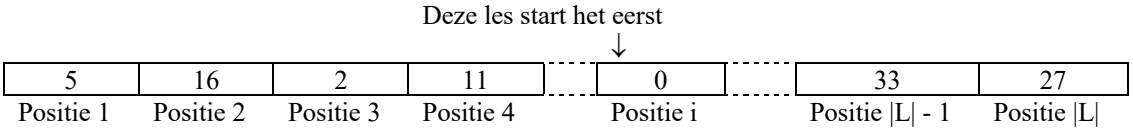
[3] Algoritme bedacht door Oliveira en Ferreira's voor oplossen van 2-dimensionale "cutting" probleem

4.3.3.1 Evolutionaire algoritme

Het algoritme is afkomstig van [18]. Voor het oplossen van het roosterprobleem wordt een standaard "elitist" evolutionaire techniek gebruikt. Het selectiecomponent is een "k-tournament" selectie met $k \geq 2$, dit is een parameter van het algoritme. De implementatie van deze selectie zorgt voor "elitism" dit houdt in dat minimaal één kopie van de beste kandidaat oplossing naar de volgende generatie gaat. Het algoritme gebruikt bij de mutatieoperatie probleemspecifieke kennis.

4.3.3.2 Kandidaat oplossing

De kandidaat oplossing is een directe codering van het probleem. Elk kandidaat oplossing uit de populatie bestaat uit een vector van positieve getallen, zoals figuur 4.7.



Figuur 4.7: Structuur van de kandidaat oplossing

De lengte van de vector is het aantal lessen die gepland moeten worden. Elk getal in een positie is de starttijd van de desbetreffende les.

4.3.3.3 Kwaliteit van de kandidaat oplossing

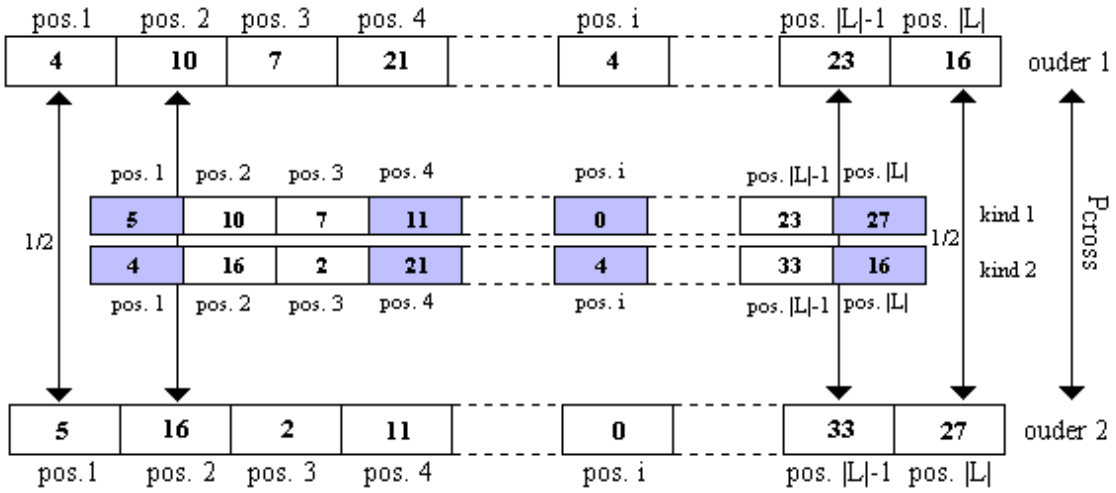
De kwaliteit van de kandidaat oplossing hangt af van het aantal voorwaarde waaraan de kandidaat oplossing niet voldoet. Er wordt een onderscheid gemaakt in harde en zachte voorwaarden. Harde voorwaarden moeten altijd voldaan worden en zachte voorwaarden moeten we zoveel mogelijk voldoen. De kwaliteit van de kandidaat oplossing wordt door de volgende formule bepaald:

$$f(g) = \frac{1}{1 + \sum_i \alpha_i h_i} + \frac{\gamma}{1 + \sum_j \beta_j s_j}$$

met $h_i \in \{0,1\}$ is de penalty gerelateerd aan de i-de harde voorwaarde en $\alpha_i \in \Re^+$ is het bijbehorende gewicht. Verder is $s_j \in \{0,1\}$ de penalty gerelateerd aan de j-de zachte voorwaarde en $\beta_j \in \Re^+$ is het bijbehorende gewicht. Tenslotte hebben we $\gamma \in \{0,1\}$ en standaard is γ nul, alleen als alle harde voorwaarden voldaan zijn dan wordt γ 1.
De gewichten, die aan iedere harde en zachte voorwaarden zijn gehecht, zijn afhankelijk van de belangrijkheid van de voorwaarde voor het desbetreffende probleem.
In deze strategie zien we dat een kandidaat oplossing eerst aan alle harde voorwaarden moet voldoen voordat de kandidaat oplossing rekening moet houden met de zachte voorwaarden. Dit betekent als alle harde voorwaarden voldaan zijn we een toegelaten oplossing krijgen.

4.3.3.4 Recombinatie

De recombinatieoperatie is een "crossover" operatie, zie figuur 4.8. Twee kandidaat oplossingen worden uit de populatie gekozen voor recombinatie. Recombinatie wordt toegepast op ieder paar van de kandidaat oplossingen met kans $P_{cross.}$ In figuur 4.8 hebben we P_{cross} is ½ genomen, maar in het algoritme is het een parameter.



Figuur 4.8: Illustratie van de "crossover" operatie

4.3.3.5 Speciale operatie

De mutatieoperatie van een standaard evolutionaire techniek is een blinde operatie. Dit houdt in dat de mutatieoperatie de kwaliteit van de kandidaat oplossing willekeurig verandert. Het gevolg is een verslechtering of een verbetering van de kwaliteit van de kandidaat oplossing.
Het algoritme past twee soorten mutatie toe, namelijk een intelligente mutatie en een verbeteringsmutatie. Bij elke generatie wordt bepaald welk van de twee mutatie wordt toegepast.

Intelligente mutatie
Deze operatie lijkt op de standaard mutatie van de evolutionaire techniek alleen zorgt deze operatie ervoor dat de kwaliteit van de kandidaat oplossing alleen verbetert. Als op positie i een verandering plaatsvindt dan heeft dit een indirect effect op de andere lessen. De intelligente mutatie zorgt ervoor dat de overtreding in de voorwaarden vermindert.

Verbeteringsmutatie
Deze operatie is een alternatief van de intelligente mutatie. De operatie zorgt voor herstructurering van de kandidaat oplossing. Eerst wordt bepaald op er gaten zijn tussen de lessen. Eventuele gaten worden opgespoord en opgevuld met lessen. De operatie zorgt vaak dat de kwaliteit van de kandidaat oplossing verbetert en zorgt ook voor een toegelaten oplossing. Deze operatie zorgt ook voor dat het algoritme niet alleen richt op lokaal zoeken.

4.3.3.6 Resultaat

Het algoritme is getest met werkelijke data uit de praktijk met ongeveer 1000 lessen. Het algoritme heeft gemiddeld 40 minuten nodig om tot een toegelaten oplossing te komen en 5 ½ uur om tot een acceptabele oplossing te komen op een 500 MHz PC. Bovenstaand algoritme is niet een standaard evolutionaire techniek maar meer een hybridische evolutionaire techniek. Zowel de intelligente mutatie als de verbeteringsmutatie gebruikt probleemspecifieke kennis.
Bovenstaand algoritme is geïmplementeerd in een commerciële software, EvoSchool[4], dat met succes op de Italiaanse markt is gelanceerd.

4.3.4 Personeelsplanning

Personeelsplanning is van groot belang voor een organisatie, omdat organisaties door een goede planning effectiever kunnen 'draaien', maar vooral ook omdat personeel daardoor op een goede en verantwoorde manier kan worden ingezet. Veel organisaties hebben te maken met personeelsplanning. In de praktijk betrekt de personeelsplanning verschillende factoren waar de planner rekening mee moet houden. Factoren, zoals ervaring en capaciteit van het personeel, vakantie regelingen, enzovoort.

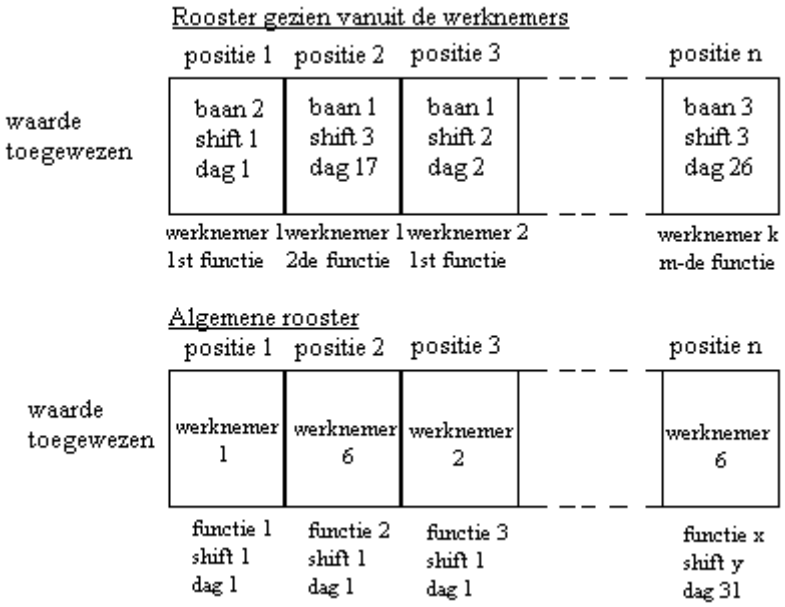
[4] Het programma is vrij te downloaden op de volgende site. http://www.genetica-soft.com/eng/evoschool.html

4.3.4.1 Evolutionaire algoritme

Het algoritme is afkomstig van [19]. De kern van het algoritme is een standaard evolutionaire techniek met "k-tournament"($k \geq 2$) selectie en "2-point crossover" met drie kandidaat oplossingen. Nieuwe kandidaat oplossingen worden aan de populatie toegevoegd. Dit gebeurt alleen als de kwaliteit van de nieuwe kandidaat oplossing beter is dan de slechtste kandidaat oplossing uit de populatie. De nieuwe kandidaat oplossing vervangt de slechtste kandidaat oplossing.

4.3.4.2 Kandidaat oplossing

De kandidaat oplossing is een directe codering van het probleem. De kandidaat oplossing stelt een maandelijkse rooster voor, zie figuur 4.9. Op iedere positie wordt een taak aan het personeel toegekend.



Figuur 4.9: Kandidaat oplossing van een maandelijkse rooster

4.3.4.3 Mutatie

De mutatieoperatie verandert willekeurig van toegewezen werknemer. Aan het begin is de mutatieratio hoog, daarna vermindert de mutatieratio lineair. Dit betekent dat aan het begin de mutatieoperatie vaker optreedt dan aan het eind. Het lineair verminderen van de mutatieratio zorgt ervoor dat het algoritme naar een optimum convergeert.

4.3.4.3 Kwaliteit van de oplossing

De kwaliteit van de oplossing is gebaseerd op penalty kosten voor iedere voorwaarden die is overtreden. Hoe lager de penalty kosten zijn hoe beter de kandidaat oplossing is. Goede kandidaat oplossing is een oplossing met weinig penalty kosten wat resulteert in een hogere kans dat het geselecteerd wordt bij de "k-tournament" selectie.

4.3.4.4 Speciale operatie

De bovenstaande beschreven strategie creëert goede kandidaat oplossingen (roosters), maar nog steeds zijn teveel voorwaarden overtreden. Om de kwaliteit en de snelheid van het algoritme te verhogen is het noodzakelijk dat probleemspecifieke kennis toegevoegd moet worden. Het idee is om speciale repareeroperaties uit te voeren. Dit gebeurt na selectie, recombinatie en mutatie maar voor de berekening van de kwaliteit van de kandidaat oplossing.
Een nadeel is dat het zoekproces zich nu meer richt op lokaal zoeken. Dus de repareer operatie mag niet te vroeg geïntroduceerd worden, anders zal het zoekproces stagneren in een bepaald gebied. De repareer operatie kan het beste geïntroduceerd worden als hun impact geleidelijk toeneemt. Een mogelijke reparatie is taken annuleren. Als een werknemer meerdere taken per dag krijgt toegewezen, dan zullen alle taken van die dag geannuleerd worden behalve 1.

4.3.4.5 Resultaat

Deze strategie is toegepast voor de personeelsplanning in het ziekenhuis. Het resultaat is in tabel 4.1. opgenomen.

Aantal werknummers	Uren per week	Aantal shifts	Aantal functies	Computer tijd
22	38,5	4	4	3 minuten
20	38,5 ; 19,25	3	5	5 minuten
24	38,5 ; 20 ; 15	2	7	5 minuten

Tabel 4.1: Drie resultaten van het algoritme

Een adaptieve versie van het algoritme is geïntegreerd in de personeelsplanning software SP-EXPERT en is sindsdien gebruikt door verschillende consumenten. Het algoritme heeft hoogstens 5 minuten (600 MHz Pentium) nodig om tot een toegelaten oplossing (rooster) te komen. De gepresenteerde techniek laat zien dat evolutionaire techniek geschikt is om het rooster probleem op te lossen. Maar het lijkt dat probleemspecifieke kennis (de repareeroperatie) nodig is om aan het eind een toegelaten rooster te creëren.

4.3.5 Productieplanning van identieke machines

Een probleem wat vaak voorkomt bij productieplanning is het toekennen van niet identieke taken aan machines. We hebben een aantal machines (machines ≥ 2) met gelijke capaciteiten en een aantal taken die uitgevoerd moeten worden. De bedoeling is een planning maken zodat de productietijd geminimaliseerd wordt. De machines kunnen per keer maar 1 taak uitvoeren en de productietijd van elke taak is bekend.

4.3.5.1 Evolutionaire algoritme

Het algoritme is afkomstig van [20]. Het algoritme is een standaard evolutionaire techniek met “elitism”. Beste kandidaat oplossingen mogen naar de volgende generatie gaan. In de algoritme is recombinatie en mutatie gebruikt. Recombinatie- en mutatieoperatie gebruiken probleemspecifieke kennis zodat de nieuwe kandidaat oplossing toegelaten blijft.

4.3.5.2 Kandidaat oplossing

De kandidaat oplossing is een directe representatie van het probleem. De structuur van de kandidaat oplossing is als volgt: <taak_id, mach_id, starttijd, eindtijd>
,met taak_id is de taak die toegewezen moet worden; mach_id is de machine die de taak uitvoert; starttijd en eindtijd is de starttijd en eindtijd van de taak van de machine. De lengte van de kandidaat oplossing is het aantal taken dat uitgevoerd moet worden. Twee voorbeelden van de kandidaat oplossingen zijn:

Kandidaat oplossing 1	1,1,0,2	2,1,2,5	3,2,2,3	4,2,3,5	5,1,5,8	6,2,5,8	7,1,8,9	8,1,9,10
Kandidaat oplossing 2	1,1,0,2	2,1,2,5	3,2,2,3	4,1,5,7	5,2,5,8	6,1,7,10	7,1,10,11	8,2,11,12

4.3.5.3 Recombinatie

Bij recombinatie is gekozen voor een “1-point crossover” operatie. Het toepassen van “1-point crossover” bij de gekozen representatie geeft een probleem. Na recombinatie kan namelijk een niet toegelaten kandidaat oplossing ontstaan. Bijvoorbeeld als we kiezen om op positie 5 de “1-point crossover” operatie toe te passen, krijgen we twee niet toegelaten kandidaat oplossingen. We krijgen dan:

Kandidaat oplossing 1	1,1,0,2	2,1,2,5	3,2,2,3	4,2,3,5	**5,1,5,8**	**6,1,7,10**	7,1,10,11	8,2,11,12
Kandidaat oplossing 2	1,1,0,2	2,1,2,5	3,2,2,3	4,1,5,7	**5,2,5,8**	**6,2,5,8**	7,1,8,9	8,1,9,10

Uit het voorbeeld kunnen we zien dat machine 1 zowel taak 5 als taak 6 op tijdstip 7 moet uitvoeren. Om dit probleem te verhelpen is gekozen voor een reparatiealgoritme. Het reparatiealgoritme repareert een niet toegelaten kandidaat oplossing tot een toegelaten kandidaat oplossing.
Het reparatiealgoritme werkt als volgt:

- Plaatst de taak waar een conflict is naar het eerst volgende tijdstip dat de machine vrijkomt;
- Starttijdstippen van taken na de conflicterende taak worden aangepast.

In ons voorbeeld, deze geavanceerde “crossover” zal de volgende kandidaat oplossingen creëren,

Kandidaat oplossing 1	1,1,0,2	2,1,2,5	3,2,2,3	4,2,3,5	5,1,5,8	6,1,8,11	7,1,11,12	8,2,11,12
Kandidaat oplossing 2	1,1,0,2	2,1,2,5	3,2,2,3	4,1,5,7	5,2,5,8	6,2,8,11	7,1,8,9	8,1,9,10

4.3.5.4 Mutatie

Bij mutatie wordt de kandidaat oplossing van links naar rechts doorlopen. Verandering kan plaatsvinden bij mach_id of bij starttijd en eindtijd. Als een taak door een vrije machine uitgevoerd kan worden dan verandert de waarde van mach_id. Ook kan een taak naar voren geplaatst worden dan veranderen de waarden starttijd en eindtijd. Deze operatie zorgt voor een toegelaten nieuwe kandidaat oplossing. Een kandidaat oplossing met mutatie:

Kandidaat oplossing 1	1,1,0,2	2,1,2,5	3,2,2,3	4,2,3,5	5,1,5,8	6,2,5,8	7,1,8,9	8,**2**,9,10

4.3.5.5 Resultaat

Het algoritme is getest met willekeurig gegenereerde data en testproblemen uit de literatuur. Goede toegelaten oplossingen zijn gegenereerd maar een optimale oplossing kan niet worden gegarandeerd. Het algoritme geeft meerdere goede oplossingen en creatie van meer dan twee nieuwe kandidaat oplossingen per koppel geeft betere resultaten dan de standaard recombinatiecomponent.

4.3.6 Insteltijden van machines

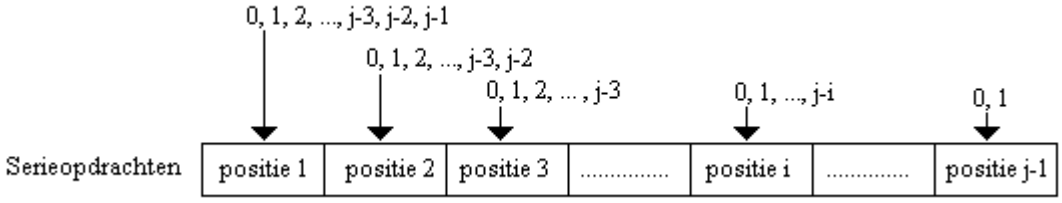
Fabrieken produceren verschillende soorten producten en de machineinstellingen voor ieder product is weer anders. Het instellen van de machine kost tijd. Tijdens de planning zal de planner rekening houden met de instelling van de machine. Producten met ongeveer dezelfde machineinstellingen zullen vlak na elkaar geproduceerd worden. Men hoeft de instelling van de machine minder te wijzigen. Het minimaliseren van de insteltijden van de machines is tevens ook het minimaliseren van de productietijd.

4.3.6.1 Evolutionaire algoritme

Het algoritme is afkomstig van [21]. De gebruikte selectiecomponent is de “ranking” selectie. Deze selectie houdt in dat de beste n kandidaat oplossingen naar de volgende generatie gaan. Verder zijn ook de recombinatie- en de mutatieoperatie toegepast. De kandidaat oplossing is een indirecte representatie van het probleem.

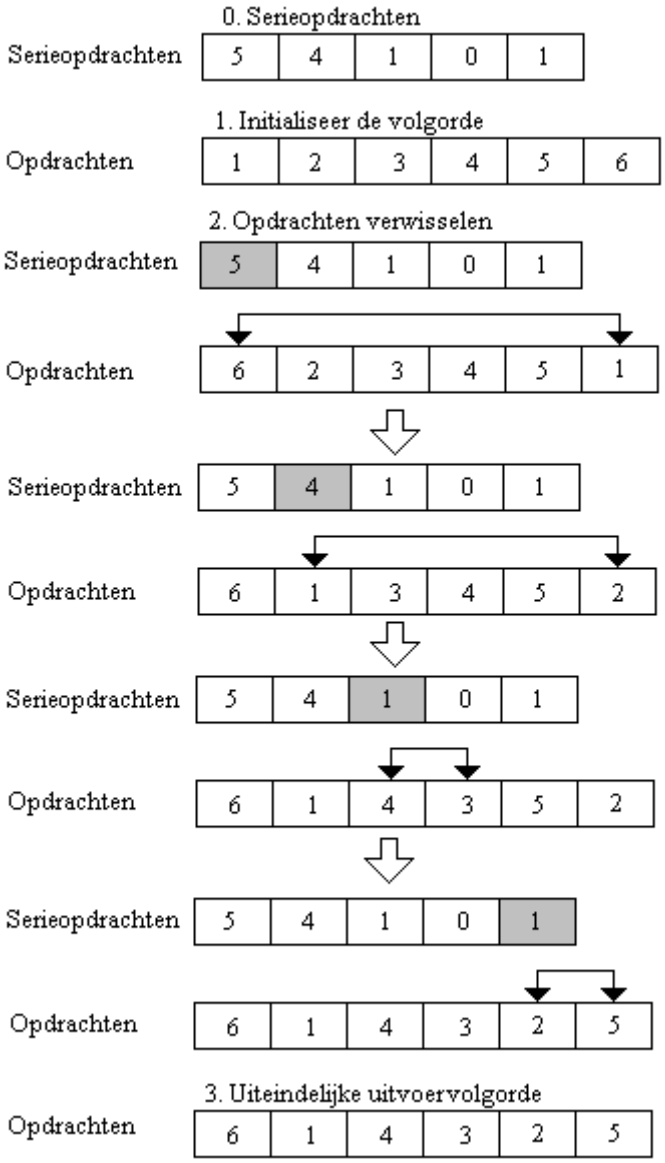
4.3.6.2 Kandidaat oplossing

De kandidaat oplossing bestaat uit een rijtje getallen met lengte j-1, j is het aantal opdrachten. Dit rijtje noemen we vanaf nu serieopdrachten en in figuur 4.10 is de structuur van de serieopdrachten opgenomen.



Figuur 4.10: Structuur van serieopdrachten

De transformatie van de serieopdrachten naar opdrachten die een machine moet uitvoeren is te zien in figuur 4.11. In figuur 4.11, de serieopdrachten zijn {5,4,1,0,1}. De uitvoervolgorde van de opdrachten is aan het begin {1,2,3,4,5,6}. De waarde op positie 1 is vijf en dit betekent dat opdracht 1 verwisseld wordt met een opdracht die 5 stappen naar rechts staat. Dus opdracht 1 en 6 zijn verwisseld en de uitvoervolgorde is nu {6,2,3,4,5,1}. Dit herhalen we voor elke positie in de serieopdrachten. De uiteindelijke uitvoervolgorde van de opdrachten wordt {6,1,4,3,2,5}.

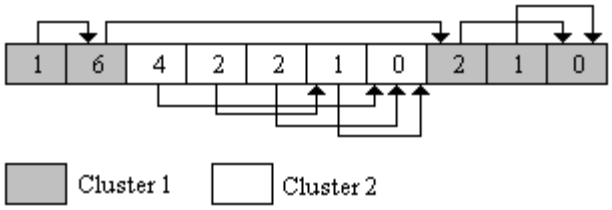


Figuur 4.11: Transformatie serieopdrachten naar uitvoervolgorde van de machine

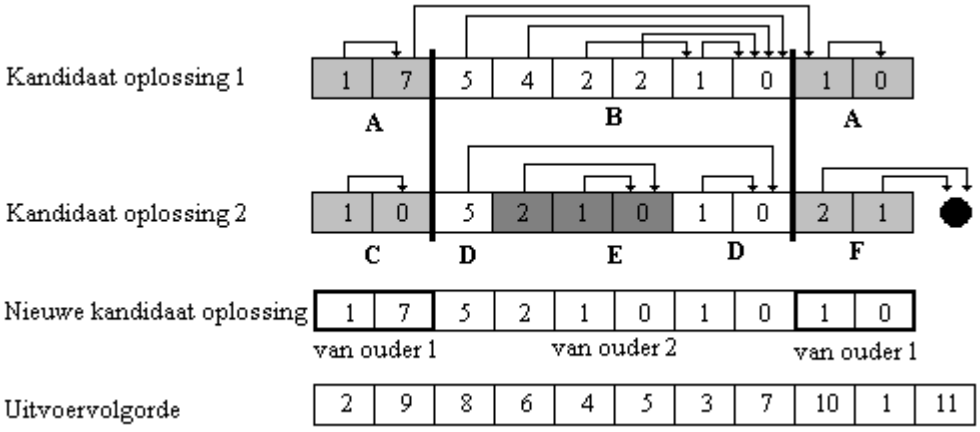
4.3.6.3 Recombinatie

Bij recombinatie operatie wordt de cluster “crossover” gebruikt. Het voordeel van deze methode is dat clusters herkend worden en de gehele cluster naar de nieuwe kandidaat oplossing gekopieerd wordt. Opdrachten die na elkaar uitgevoerd worden, zullen bij de nieuwe kandidaat oplossing ook na elkaar uitgevoerd worden. In figuur 4.12 is te zien hoe zo cluster ontdekt wordt en in figuur 4.13 is te zien hoe zo een recombinatie plaatsvindt.

Een nieuwe kandidaat oplossing bestaat uit een gedeelte van kandidaat oplossing 1 en een gedeelte van kandidaat oplossing 2. Kandidaat oplossing 1 heeft twee clusters, A en B. Kandidaat oplossing 2 heeft vier clusters C, D, E en F. Omdat twee grenzen van kandidaat oplossing 1 en 2 overeenkomen, kunnen clusters op de grenzen verwisseld worden.



Figuur 4.12: De clusters van subvolgorde



Figuur 4.13: Cluster "Crossover"

Er zijn twee gevallen waarbij cluster "crossover" niet werkt. Het eerste geval gebeurt als er geen grenzen zijn waar beide ouders overeenkomen. Het tweede geval gebeurt als de kandidaat oplossing maar uit 1 cluster bestaat. Het eerste geval lossen we op door alleen rekening te houden met de clusters van één van de kandidaat oplossing, dit noemen we de quasi-cluster "crossover". De clusters van de andere kandidaat oplossing negeren we. Het tweede geval lossen we op door "1-point crossover" toe te passen. Deze oplosmethode houdt geen rekening met subvolgorden.

4.6.3.4 Mutatie

De mutatieoperatie is vrij simpel. De mutatie verandert de waarde bij de serieopdrachten op een bepaalde positie willekeurig in een ander getal. Het getal is gekozen uit een homogene[0, j-i] verdeling.

4.6.3.5 Resultaat

Het algoritme is getest met testproblemen uit de literatuur en data uit de praktijk. Een interessant resultaat zijn de drie verschillende recombinaties. In de eerste generatie is 30% van de recombinatieoperatie de standaard cluster "crossover", 55% quasi-cluster "crossover" en 15% "1-point crossover". Het percentage van de standaard cluster "crossover" stijgt geleidelijk. In de laatste generatie is 80% van de recombinatieoperatie de standaard cluster "crossover", 15% quasi-cluster "crossover" en 5% "1-point crossover".
De techniek is toegepast op een fabrikant die geluidsonderdelen produceert. De fabrikant produceert 7500 producten per maand. Het algoritme moet dus voor de leverancier een maandelijkse planning maken. De fabrikant gebruikt een planningsprogramma om de planning te maken. De uitvoer van het planningsprogramma moet handmatig door een planner gewijzigd worden om de insteltijd te verkorten. Het handmatig optimaliseren van de planning kost de planner drie tot vier uur.
Uit de resultaten blijkt dat de evolutionaire techniek een beter planning produceert dan het planningspakket die de fabrikant nu gebruikt. Doordat de insteltijden verkort zijn, is het aantal vertraagde leveringen ook verminderd.

4.3.7 Seriegewijze productie

Het vervaardigen van een product in een fabriek is een doorlopend proces. De input van de fabriek zijn vaak grondstoffen of halffabrikaten en de output is het eindproduct. Om van grondstoffen een eindproduct te vervaardigen moeten de grondstoffen een aantal bewerkingen ondergaan. Deze bewerkingen moeten op de juiste volgorde uitgevoerd worden. Men noemt dit de seriegewijze of batch-gewijze productie.

4.3.7.1 Evolutionaire algoritme

Het algoritme is afkomstig van [22]. Het algoritme past "elitism" toe. De beste kandidaat oplossing van ieder generatie gaat altijd naar de volgende generatie. Nieuwe kandidaat oplossingen gaan automatisch naar de volgende generatie en vervangen willekeurig oude kandidaat oplossingen. Voor recombinatie gebruiken we de twee beste kandidaat oplossingen. Bij mutatie, het beste van de oude kandidaat oplossing en de gemuteerde kandidaat oplossing gaat naar de volgende generatie. Bij iedere generatie komen er twee of drie nieuwe kandidaat oplossingen bij en twee of drie oude kandidaat oplossingen worden vervangen door de nieuwe kandidaat oplossingen.

4.3.7.2 Kandidaat oplossing

De structuur van de kandidaat oplossing is een string van N x M posities, met N taken en M machines.
Elk positie representeert een operatie:

$$[\text{operatie}_1 \ \ldots \ \text{operatie}_i \ \ldots \ \text{operatie}_{NxM}]$$

Een operatie is het uitvoeren van een taak op een machine.

4.3.7.3 Recombinatie

Bij recombinatie is het belangrijk dat de machinevolgordematrix gegeven is. De matrix laat de uitvoervolgorde van de taken zien. Als voorbeeld gebruiken we de volgende machinevolgorde N x M matrix:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 4 & 2 \\ 1 & 3 & 2 & 4 \\ 3 & 2 & 4 & 1 \end{bmatrix}$$

Dus taak 2 moet achtereenvolgens door machine 3, 1, 4 en 2 uitgevoerd worden.

Tijdens de recombinatie worden eerst twee kandidaat oplossingen geselecteerd. De eerste positie van kandidaat oplossing 1 betekent dat taak 2 door machine 3 wordt uitgevoerd.

Kandidaat oplossing 1: $2_3\ 2_1\ 3_1\ 4_3\ 3_3\ 1_1\ 4_2\ 3_2\ 1_2\ 1_3\ 2_4\ 2_2\ 1_4\ 4_4\ 3_4\ 4_1$
Kandidaat oplossing 2: $1_1\ 1_2\ 4_3\ 3_1\ 1_3\ 3_3\ 2_3\ 4_3\ 4_4\ 2_1\ 1_4\ 2_4\ 2_2\ 3_2\ 4_1\ 3_4$

Nu selecteren we willekeurig k($1 \leq k \leq N-1$) taken, bijvoorbeeld taak 1 en 3. Op de lege posities zetten we een H neer. Kandidaat oplossing 1 wordt dan

Kandidaat oplossing 1: H H 3_1 H 3_3 1_1 H 3_2 1_2 1_3 H H 1_4 H 3_4 H

Niet geselecteerde operaties van kandidaat oplossing 1 zijn

$2_3\ 2_1\ 4_3\ 4_2\ 2_4\ 2_2\ 4_4\ 3_4\ 4_1$

Voor kandidaat oplossing 2 krijgen we

Kandidaat oplossing 2: $1_1\ 1_2$ H $3_1\ 1_3\ 3_3$ H H H H 1_4 H H 3_2 H 3_4

Niet geselecteerde operaties van kandidaat oplossing 2 zijn

$4_3\ 2_3\ 4_3\ 4_4\ 2_1\ 2_4\ 2_2\ 4_1\ 3_4$

De volgende stap is het verschuiven van de string L posities voorwaarts of achterwaarts. De L posities zijn willekeurig bepaald en de afstand tussen de verschillende posities blijft gelijk. Als voorbeeld nemen we 1 positie voorwaarts voor kandidaat oplossing 1, we krijgen dan:

String 1: H 3_1 H 3_3 1_1 H 3_2 1_2 1_3 H H 1_4 H 3_4 H H

Bij kandidaat oplossing 2 is geen verschuiving mogelijk dus nemen we L is 0, we krijgen dan:

String 2: 1_1 1_2 H 3_1 1_3 3_3 H H H H 1_4 H H 3_2 H 3_4

De laatste stap is om de lege posities op te vullen met niet geselecteerde operaties. String 1 wordt opgevuld met niet geselecteerde operaties van kandidaat oplossing 2. Bij string 2 vullen we de lege posities met niet geselecteerde operaties van kandidaat oplossing 1. We krijgen de volgende nieuwe kandidaat oplossingen:

Nieuwe kandidaat oplossing 1: 4_3 3_1 2_3 3_3 1_1 4_2 3_2 1_2 1_3 4_4 2_1 1_4 2_4 3_4 2_2 4_1
Nieuwe kandidaat oplossing 2: 1_1 1_2 2_3 3_1 1_3 3_3 2_1 4_3 4_2 2_4 1_4 2_2 4_4 3_2 4_1 3_4

4.3.7.4 Mutatie
We passen de mutatie toe op kandidaat oplossing 1.

Kandidaat oplossing 1: 2_3 2_1 3_1 4_3 3_3 1_1 4_2 3_2 1_2 1_3 2_4 2_2 1_4 4_4 3_4 4_1

Eerst wordt een taak willekeurig geselecteerd. Als voorbeeld nemen we taak 1 en we krijgen dan

Kandidaat oplossing 1: H H H H H 1_1 H H 1_2 1_3 H H 1_4 H H H

De niet geselecteerde operaties zijn:

2_3 2_1 3_1 4_3 3_3 4_2 3_2 2_4 2_2 4_4 3_4 4_1

Daarna herplaatsen we de geselecteerde taak willekeurig. Wel moeten we rekening houden dat de afstand tussen de operaties gelijk blijft. Als voorbeeld plaatsen we operatie 1_1 op positie 3 (in ons voorbeeld, de nieuwe positie kan gekozen worden uit 1 tot en met 9). We krijgen dan:

Kandidaat oplossing 1: H H 1_1 H H 1_2 1_3 H H 1_4 H H H H H H

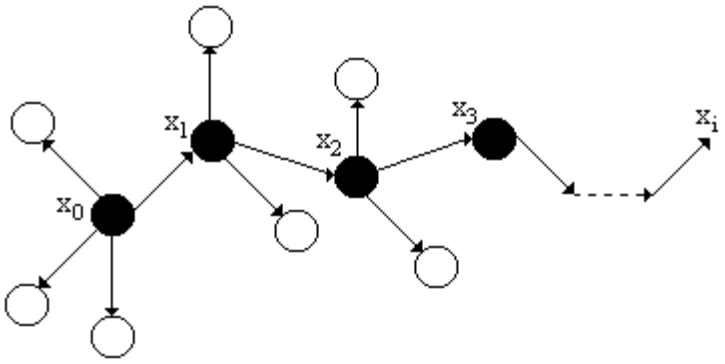
We vullen daarna de lege positie met de niet geselecteerde operaties in dezelfde volgorde. We krijgen dan

Gemuteerde kandidaat oplossing 1: 2_3 2_1 1_1 3_1 4_3 1_2 1_3 3_3 4_2 1_4 3_2 2_4 2_2 4_4 3_4 4_1

4.3.7.5 Speciale operatie
De speciale operatie is een lokaal zoekmethode. De methode zoekt de beste oplossing van een kleine omgeving. Een oplossing x representeert een punt in de zoekruimte en zijn buren N(x) zijn gedefinieerd als een set van toegelaten oplossingen. Oplossing x moet wel N(x) binnen 1 stap kunnen bereiken. Voor het min f(x), de zoekmethode is als volgt:

Lokaal zoekmethode:
(1) Selecteer een startpunt x_i, i = 0.
(2) Bepaal de buren $N(x_i)$ van x_i en evalueer $f(x_i)$, $x_i \in N(x_i)$.
(3) Zoek x* van $\{x_i, N(x_i)\}$ volgens: $f(x^*) = \min_{z \in \{x_i, N(x_i)\}} f(z)$.
(4) Laat i = i + 1, $x_i = x^*$, herhaal (2) en (4) totdat de eindconditie is bereikt.

In figuur 4.14 is de lokaal zoekmethode opgenomen.



Figuur 4.14: Lokaal zoekmethode

In het algoritme, de buren van een kandidaat oplossing is gedefinieerd als de verzameling van alle toegelaten kandidaat oplossingen geproduceerd door verwisseling van twee naast elkaar liggende posities. Het schema voor het algoritme ziet er als volgt uit:
(1) Initialiseer de populatie.
(2) Doe recombinatie op geselecteerde kandidaat oplossingen, pas lokaal zoeken op de nieuwe kandidaat oplossing toe.
(3) Doe mutatie op geselecteerde kandidaat oplossingen, pas lokaal zoeken op de nieuwe kandidaat oplossing toe.
(4) Herproduceer een nieuwe populatie.
(5) Herhaal (2) en (4) totdat de eindconditie is voldaan.

4.3.7.6 Resultaat

Het algoritme is getest met behulp van drie testproblemen uit de literatuur. Voor de drie testproblemen weten we het globale optimum. Het algoritme kan voor alle drie de problemen de optimale planning vinden. De nieuwe recombinatieoperatie is bewezen nuttig te zijn. Het versnelt het zoekproces en garandeert dat de optimale oplossing gevonden wordt. Het algoritme verkrijgt een beter convergentie door het introduceren van een lokaal zoekproces.

4.3.8 Positioneren van vliegtuigen

Optimaal positioneren van een vliegtuig bij een specifieke luchthaven is een heel moeilijk probleem. Dit komt doordat we ons aan verschillende beperkingen moeten houden. Verder moeten we ook rekening houden met de verschillende verzoeken, zoals vliegtuigtype, verzoeken van luchtvaartmaatschappijen, speciale poorten en beveiligingsbeperkingen. De hoofdtaak van vliegtuigpositionering is een optimale planning vinden voor inkomende en uitgaande vluchten met betrekking tot hun positie bij de overeenkomstige poorten van een luchthaven.

4.3.8.1 Evolutionaire algoritme

Het algoritme is afkomstig van [23]. De gebruikte strategie om het bovenstaande probleem op te lossen is niet bekend. Wel is bekend dat ze zowel recombinatie- en mutatieoperaties gebruiken. Doordat we rekening moeten houden met verschillende beperkingen is de algemene recombinatie- en mutatieoperaties niet voldoende. Voor beide operaties moet probleemspecifieke kennis gebruikt worden.

4.3.8.2 Kandidaat oplossing

De structuur van de kandidaat oplossing is te zien in figuur 4.14.

Positie:	1	2	3	4	5	6	7	...
Vliegtuig:	3		1		2	5	6	...
					4		7	

Figuur 4.14: Structuur van de kandidaat oplossing

Op positie vijf zijn twee vliegtuigen gepositioneerd.

4.3.8.3 Mutatie

Bij de mutatieoperatie is gekozen om drie verschillende mutatieoperaties toe te passen.

Mutatie 1

Mutatie 1 is de simpelste mutatie operatie. De operatie zoekt alle vliegtuigen op die op de verkeerde positie staan. Bijvoorbeeld een B747 kan alleen op positie 1, 2 en 3 staan, maar als die op positie 5 staat moet die verplaatst worden naar 1, 2 of 3. Deze operatie zorgt ervoor dat alle vliegtuigen op een positie staan waar ze mogen staan.

Mutatie 2

Deze mutatieoperatie wordt gebruikt om de verscheidenheid in de populatie te handhaven. De mutatieoperatie verplaatst een vliegtuig van 1 positie naar een andere positie. Deze operatie gebeurt in drie stappen. De eerste stap is willekeurig selecteren van een vliegtuig. Dit vliegtuig gaat naar een tijdelijke positie, alle vliegtuigen in de tijdelijke positie zijn nog niet gepositioneerd. De tweede stap is het willekeurig selecteren van een positie, het vliegtuig moet wel in deze positie passen. De derde stap is het vliegtuig op de geselecteerde positie te plaatsen. Als er al een vliegtuig op de geselecteerde positie staat dan wordt dit vliegtuig naar de tijdelijke positie verplaatst. Stap 2 en 3 worden herhaald totdat er zich geen vliegtuigen meer in de tijdelijke positie bevinden.

Mutatie 3

Deze mutatieoperatie werkt net als mutatieoperatie 2. Deze mutatie werkt niet op één vliegtuig, maar op complete posities.

4.3.8.4 Recombinatie

We gebruiken geen standaard recombinatietechnieken omdat ze bij deze technieken geen rekening houden met clusters. Clusters zijn groepsdelen in de kandidaat oplossing die we zoveel mogelijk bij elkaar willen houden. Er zijn drie soorten recombinatieoperaties ontwikkeld.

Recombinatie 1

Deze recombinatieoperatie gedraagt zich als een normale recombinatiefunctie. Vliegtuigen die na recombinatie verkeerd staan, worden gecorrigeerd net als bij mutatie 1.

Recombinatie 2

Deze recombinatie gebruikt veel probleemspecifieke kennis. Het idee is om een overdracht te maken van perfect gepositioneerde vliegtuigen. Perfect gepositioneerde vliegtuigen en hun positie worden gekopieerd naar de nieuwe kandidaat oplossing. Een set van regels bepaalt of een vliegtuig perfect gepositioneerd is.

Recombinatie 3

Deze recombinatie lijkt veel op recombinatie 2, alleen wordt er nu niet gekeken naar 1 positie maar naar meerdere posities tegelijk. Een groep perfect gepositioneerde vliegtuigen en zijn posities worden gekopieerd naar de nieuwe kandidaat oplossing.

4.3.8.5 Kwaliteit van de oplossing

De doelfunctie F bestaat uit verschillende bestandsdelen. F(i) is gebaseerd op vier hoofd optimalisatiecriteriums met betrekking tot kandidaat oplossing i:

$$F(i) = P(i) + C(i) + S(i) + Q(i)$$

P(i) is het onderdeel dat probeert het aantal passagiers te optimaliseren. C(i) is verantwoordelijk voor de vermindering van de verbindingstijd. S(i) beschrijft de serviceaspecten en Q(i) de kwaliteit van de oplossingen. Alle vier onderdelen bestaan weer uit verschillende bestanddelen waar we hier niet verder op ingaan.

4.3.8.6 Resultaat

Het systeem is getest op een echte scenario van de Frankfurt's luchthaven. De verkregen resultaten van het systeem zijn geëvalueerd en het systeem produceert betere oplossingen dan menselijke experts. Een expert heeft ongeveer een week nodig om een optimale planning te produceren. Het systeem heeft een uur nodig voor dezelfde taak en de geproduceerde planning is nog beter dan de menselijke expert.

Hoofdstuk 5 Conclusie en vervolg onderzoek

5.1 Inleiding

Wereldwijd is momenteel veel interesse in evolutionaire techniek. Deze interesse wordt veroorzaakt door de ontoereikendheid van lineaire programmering in het oplossen van complexe optimalisatieproblemen. Complexe optimalisatieproblemen kunnen vaak niet meer opgelost worden met analytische methoden, oplossen met dergelijke methoden kosten doorgaans veel reken- of computertijd. Voor heuristische methoden (evolutionaire techniek) geldt dit in minder mate. Hierbij staat een acceptabele oplossing, die in de buurt komt van de optimale oplossing, binnen een redelijke reken- of computertijd op de voorgrond. Van de aldus resulterende bruikbare oplossingen kan echter niet – zoals bij de analytische methoden – worden gezegd dat deze oplossing wiskundig gezien optimaal is. Voor veel vraagstukken in de praktijk vormt dit geen belemmering. Dit merken we ook op aan de hand van de gepresenteerde voorbeelden (reclameprobleem paragraaf 4.2.3, capaciteitsprobleem van vrachtwagens paragraaf 4.3.1, personeelsplanning paragraaf 4.3.4 en insteltijden van machines paragraaf 4.3.6) in dit werkstuk.
Het hoofdstuk is als volgt opgebouwd: in paragraaf 5.2 concluderen we het toepassen van evolutionaire techniek bij optimalisatie van bedrijfsprocessen en in paragraaf 5.3 geven we interessante vervolg onderzoeksgebieden aan.

5.2 Conclusie

Evolutionaire technieken zijn veelbelovende en breed toepasbare zoek- en optimalisatietechnieken, gebaseerd op het mechanisme van natuurlijke evolutie. De techniek heeft zich bewezen als een van de belangrijkste technologieën voor het bouwen van adaptieve systemen. Het systeem is instaat om met complexe en veranderde omgevingen om te gaan, net zoals bij de natuurlijke evolutie. Interesse in evolutionaire techniek is in de laatste jaren dramatische toegenomen en de techniek is met succes toegepast op verscheidene complexe problemen.
De gepresenteerde optimalisatievoorbeelden in dit werkstuk geven aan dat voor het toepassen van evolutionaire techniek men rekening moet houden met verschillende problemen, zoals de representatie, evolutionaire componenten en strategische parameters. De structuur van de kandidaat oplossing hangt af van het probleem. Een goed gekozen structuur leidt vaak tot een kortere evolutie tijd. Behalve de structuur van de kandidaat oplossing is het ook van belang om de geschikte evolutionaire componenten te kiezen. Niet alle componenten zijn geschikt voor alle structuren. Voor de bepaling van de kwaliteit van de kandidaat oplossing hebben we een doelfunctie nodig. In de praktijk zal het ontwikkelen van een doelfunctie in samenwerking moeten gaan met experts van het desbetreffende toepassingsgebied. Een ander belangrijk aandachtspunt zijn de strategische parameters. Het zoeken naar goede waarden voor de parameters (populatiegrootte, mutatieratio, etc) is soms moeilijk en vereist ervaring.
Uit de gepresenteerde optimalisatie voorbeelden kunnen we opmaken dat alleen het toepassen van de standaard evolutionaire techniek niet voldoende is. Er is geen garantie dat het algoritme binnen een beperkte tijd het optimum kan vinden. Verder heeft de evolutionaire techniek aan het eind veel tijd nodig voor het optimaliseren van de kandidaat oplossing. Probleemspecifieke kennis is toegevoegd aan het algoritme om bovenstaande problemen op te kunnen lossen.
Concluderend, evolutionaire techniek is een veelbelovende techniek, toepasbaar op vele complexe optimalisatieproblemen. Integratie van evolutionaire techniek met probleemspecifieke kennis heeft grote potentie voor praktische applicaties op het gebied van operationele research. Hoewel het ontwikkelen van hybridische evolutionaire techniek systemen meer kennis en middelen vereist, levert het systeem een uitstekende performance op, door de combinatie van verschillende benaderingen in een geïntegreerd systeem.

5.3 Vervolg onderzoek

In dit werkstuk hebben we ons onderzoek beperkt tot het gebied van operationele research. Uit de praktijk blijkt dat evolutionaire techniek veel succes boekt op verschillende gebieden, zoals op financieel gebied, engineering en ontwerpproblemen. Een interessant vervolg onderzoeksgebied zijn classificatiesystemen. Het onderzoek richt zich op toepassing van evolutionaire techniek in classificatiesystemen. Uit de literatuur blijkt dat evolutionaire techniek op dit gebied redelijk veel succes boekt. Interessant is de gebruikte evolutionair algoritme bij deze systemen.

Ter afsluiting,
What limit can we put to this power, acting during long ages and rigidly scrutinizing the whole constitution, structure and habits of each creature – favoring the good and rejecting the bad? I can see no limit to this power in slowly and beautifully adapting each form to the complex relations of life.
- Charles Darwin, *On the Origin of Species*

Bijlage 1. Literatuurlijst

[1] A.E. Eiben, Evolutionary computing: the most powerful problem solver in the universe?, Dutch Mathematical Archive (Nederlands Archief voor Wiskunde), vol. 5/3, nr. 2, 126-131, 2002

[2] Z. Michalewicz. Genetic Algorithms + Data structures = Evolution programs. Springer, 3th edition, 1996

[3] D. Whitley. The GENITOR Algorithm and Selection Pressure: Why Rank-Based allocation of reproductive trails is the best. In Schaffer[130], pp. 116-121

[4] Georges R. Harik, Finding Multimodal Solutions Using Restricted Tournament Selection, Proceedings of the Sixth International Conference on Genetic Algorithms, Morgan Kaufmann, pp. 24-31, 1995

[5] K. A. De Jong and W. M. Spears, An analysis of the interacting roles of population size and crossover in genetic algorithms, Parallel Problem Solving from Nature - Proceedings of 1st Workshop, {PPSN} 1, volume = 496, month = 1-3, Springer-Verlag, Berlin, Germany, pp. 38-47, 1991

[6] G. Rudolph, Convergence properties of canonical genetic Algorithms, IEEE Transactions on Neural Networks, 5(1): 96-101, 1994

[7] Claude McMilland Jr, Mathematical Programming 2nd Edition, ISBN: 0-471-58572-6

[8] Katty Marty, Linear and Combinatorial Programming, ISBN: 0-471-57370-1

[9] L.R. Foulds, Optimization techniques, pp. 187-194, ISBN: 0-387-90586

[10] Hans G. Daellenbach, John A. George, Introduction to Opterations Research Techniques, ISBN: 0-205-05755-1

[11] K.J. Sijsling, Technieken van Operations Research, 3de Editon, ISBN: 90-5261-226-9

[12] http://www.ncs.co.uk/solving/recipe.htm

[13] http://www.nutechsolutions.com/pdf/cs_esprit_supply_chain.pdf

[14] http://www.attar.com/pages/dev_gap.htm

[15] http://www.nutechsolutions.com/pdf/cs_siemens_network.pdf

[16] http://www.nutechsolutions.com/pdf/cs_dutch_min_traffic.pdf

[17] Jörg Biethahn, Volker Nissen, Evolutionary Algorithms in Management Applications, ISBN: 3-540-60382-4

[18] Calogero Di Stefano , Andrea G. B. Tettamanzi, An Evolutionary Algorithm for Solving the School Time-Tabling Problem, Springer Volume 2037, pp 0452

[19] Matthias Gröbner , Peter Wilke, Optimizing Employee Schedules by a Hybrid Genetic Algorithm, Springer Volume 2037, pp 0463

[20] Susana Esquivel , Claudia Gatica , Raul Gallard, Conventional and Multirecombinative Evolutionary Algorithms for the Parallel Task Scheduling Problem, Springer Volume 2037, pp 0223

[21] Tezuka Masaru , Hiji Masahiro , Miyabayashi Kazunori , Okumura Keigo, A New Genetic Representation and Common Cluster Crossover for Job Shop Sheduling, Problems, Springer Volume 1803, pp 297-306

[22] L.W. Cai , Q.H. Wu , Z.Z. Yong, A Genetic Algorithm with Local Search for Solving Job Shop Problems, Springer Volume 1803, pp 107-116

[23] J. Pfalzgraf , K. Frank , J. Weichenberger , S. Stolzenberg, Design, Implementation, and Application of a Tool for Optimal Aircraft Positioning, Springer Volume 1803, pp 382-393