2005:233 SHU

EXAMENSARBETE

Genetiska algoritmer

Peter Engberg

Luleå tekniska universitet

Systemvetenskap
D-nivå
Institutionen för Industriell ekonomi och samhällsvetenskap
Avdelningen för Systemvetenskap

2005:233 SHU - ISSN: 1404-5508 - ISRN: LTU-SHU-EX--05/233--SE

Abstract

This composition investigates how genetic algorithms are descibed in theory and how they are employed against a specific otimazationproblem. In order to achieve the aim of the report, I have chosen to perform a complete document-study, in which I have concentrated on how genetic algorithms are employed against the traveling salesman problem.

All collected material have been analysed and I have identified and described a production planning problem. The problem has been approached with genetic algoritms by developing a model that shows how genetic algorithms could be used to attack my chosen production planning problem.

The study of genetic algorithms and the practical work, have given me the experience on how genetic algorithms work and overview of the factors that are important during the development and design of a genetic algorithms. From this perspective I will answer the research question of this report: what is important to think about during development and design of genetic algorithms?

During writing this paper have I met several enthralling discoveries concerning genetic algorithms and I want through this report facilitate for those that work with development and design of genetic algorithms by introducing my personal discoveries from working with genetic algorithms.

Sammanfattning

Denna D-uppsats undersöker hur genetiska algoritmer beskrivs i teorin och hur de tillämpas mot verklighetsbaserade optimeringsproblem. I uppsatsen har jag valt att utföra en dokumentstudie, där jag undersöker hur genetiska algoritmer tillämpas mot the traveling salesman problem. Allt insamlat material har analyserats och jag har identifierat och beskrivit ett produktionsproblem, vilket jag har angripit genom att egenutveckla en modell över en genetisk algoritm.

Mina studier av genetiska algoritmer och det praktiska arbetet med att utveckla en modell över en genetisk algoritm har gett mig erfarenhet av hur en genetisk algoritm fungerar och vad som är viktigt att beakta vid utveckling och design av en genetisk algoritm. Utifrån dessa förutsättningar besvarar jag uppsatsens forskningsfråga, nämligen vad är viktigt att beakta i samband med utveckling och design av genetiska algoritmer?

Under arbetet med denna uppsats har jag uppmärksammat flera intressanta företeelser runt omkring genetiska algoritmer. Jag vill med denna uppsats underlätta arbetet för den som utvecklar och designar genetiska algoritmer genom att presentera mina personliga erfarenheter med att arbeta med genetiska algoritmer.

Förord

Den här uppsatsen är ett resultat av ett examensarbete på D-nivå i systemvetenskap utförd vid institutionen för industriell ekonomi samhällsvetenskap (IES) vid Luleå tekniska universitet (LTU).

Denna uppsats har behandlat genetiska algoritmer och dess användningsområden inom datoriserade beslutsstödjande system. Jag hoppas att uppsatsen kan ge läsaren en insikt i vad genetiska algoritmer är, hur det fungerar och även besvara frågor, rörande vad genetiska algoritmer kan användas till.

Jag vill tacka alla inblandade i mitt uppsatsskrivande, så väl som handledare, studenter och åhörare som gett tips, kritik och förslag på förbättringar och på detta sätt hjälpt mig genom denna period. Ni har hjälpt till att forma uppsatsen och räddat mig från diverse felsteg.

Luleå, Januari 2005

Peter Engberg

Innehållsförteckning

1 Inledning

I detta inledande kapitel kommer jag i tur och ordning att ge en bakgrund för uppsatsen, beskriva problemområdet och motivera den ställda forskningsfrågan, beskriva syftet med uppsatsen, förklara de avgränsningarna jag valt att göra och slutligen beskriva hela arbetsgången för uppsatsen med hjälp av en illustration.

1.1 Bakgrund

Världen är full av olika typer av problemområden. Ekonomiska problem, designproblem och matematiska problem är bara ett fåtal exempel. Problemlösning har länge varit ett ämne som fascinerat människan och på senare tid, då datorer blivit allt mer lättillgängliga och finns i var mans hem, tar människan allt oftare hjälp av datorer för att söka konsultering vid problemlösning. Beslutstödjande system är en typ av system vars främsta uppgift är att hjälpa användarna att ta beslut i problemsituationer (Marakas G., 2003).

Vid avslutningen ”Intelligenta beslutstödjande system (IBSS)” vid Luleå Tekniska Universitet (LTU) talas det om att skapa intelligenta system, som kan underlätta för beslutstagare vid olika beslutssituationer. Ett intelligent system skiljer sig en hel del från traditionella applikationer, som oftast är uppbyggda av färdiga regler som konsulteras och endast kan fungera i situationer då all efterfrågad data existerar och är komplett.

För att uppnå en viss intelligens i dessa system talas det om olika metoder för att skapa intelligenta system. Det hela kan liknas med det vi i dagligt tal brukar sammanfatta under begreppet ”artificiell intelligens”. Detta område är fortfarande i sin linda och forskare världen över arbetar inom olika områden, exempelvis neurala nätverk[1], genetiska algoritmer, fuzzy logics[2] och hybrider av olika slag. Alla dessa metoder har sina specifika karaktärsdrag och går att tillämpa mot olika typer av problem, där vissa metoder lämpar sig bättre på vissa typer av problem (Juang C., 2004).

Genetiska algoritmer är ett koncept som bygger på evolution och naturligt urval enligt darwinismen (Leou J. & Lin D., 1997). En algoritm är en samling instruktioner för att lösa ett problem. Genetiska algoritmer bygger på funktioner inspirerade av funktioner i naturen. Bättre och bättre lösningar skapas utifrån tidigare generationer, tills ett tillfredställande resultat har uppnåtts. (Aronson J. & Turban E., 2001) Exempel på detta är om en användare skall besöka fyra städer och vill färdas genom dessa städer, en gång och endast en gång, samtidigt som den totala färdsträckan skall bli så kort som möjligt. Den genetiska algoritmen hittar den kombination av de fyra städerna som ger den

[1] Neurala nätverk är en modell som försöker efterlikna mänskliga biologiska neurala nätverk och de processer som tar plats i den mänskliga hjärnan. Aronson J. & Turban E. (2001)

[2] Fuzzy logics är en metod som används för att hantera osäkerhet i datoriserade miljöer. Denna metod simulerar mänskliga processer, i syfte att tillåta en dator att bete sig mindre exakt och logisk än en dator gör i vanliga fall. Aronson J. & Turban E. (2001)

kortaste ressträckan, genom att generera lösningar och utvärdera dessa i ett repetitivt mönster. Problemet som beskrivs ovan kallas även the traveling salesman problem (Harmel B. & Patterson M., 2003) och förklaras närmare i kapitel 2.6.

1.2 Problemområde

Genetiska algoritmer har tillämpats vid inlärning av styrprogram inom robotik (Nordin P. & Wilde J., 2003), för att lösa the traveling salesman problem (TSP) (Li G. & Louis S., 2000) och vid design av datoriserade nätverks topologier (Legault G. & Pierre S., 1998), vilket alla är exempel på problemområden där genetiska algoritmer har använts.

Ett beslutstödjande system arbetar ofta inom ett specifikt problemområde med specifika arbetsuppgifter. För att genomföra och skapa ett fungerande beslutstödjande system, behövs det någon teknik som behandlar det aktuella problemet. (Marakas G., 2003) Val av teknik är ofta väldigt problemberoende och inom många problemområden finns noggrant utarbetade metoder för att arbeta fram ett godtagbart resultat. Det finns ett behov för en användare att behärska en lämplig teknik, som är tillämpbar mot det aktuella problemet användaren står inför.

Marakas G., (2003) kategoriserar beslutsstödjande system i sju grupper. Jag har studerat litteratur och tillämpningar av genetiska algoritmer och sett att genetiska algoritmer ofta förekommer i samband med optimeringsproblem, vilket även är en av de grupper av beslutstödjande system som Marakas G., (2003) nämner. Utifrån denna bakgrund vill jag undersöka hur genetiska algoritmer fungerar och om dessa kan ses som en lämplig teknik för att angripa optimeringsproblem. Jag vill genom denna uppsats dela med mig av mina erfarenheter av genetiska algoritmer till uppsatsens läsare.

1.3 Forskningsfråga

Vad är viktigt att beakta i samband med utveckling och design av genetiska algoritmer?

1.4 Syfte

Med denna uppsats vill jag ge läsaren en inblick i vad genetiska algoritmer är och hur de fungerar, men framförallt vill jag dela med mig av mina erfarenheter av genetiska algoritmer. Jag vill ge en inblick över vad genetiska algoritmer kan tillföra en utvecklare, som arbetar med datoriserade beslutsstödjande system inom kategorin optimering, men även vad utvecklaren av algoritmen bör beakta under själva utvecklingsfasen av algoritmen.

Detta vill jag åstadkomma genom att studera litteratur och utföra en dokumentstudie över beskrivna tillämpningar av genetiska algoritmer mot optimeringsproblem. Detta gör jag i syfte att kunna presentera en bild över hur genetiska algoritmer fungerar och tillämpas inom sitt specifika problemområde. Därefter avser jag att identifiera ett

optimeringsproblem och utveckla en modell, med vilket jag vill visa hur genetiska algoritmer kan angripa det aktuella problemet. Detta utförs i syfte att förklara och bevisa hur genetiska algoritmer kan fungera mot ett optimeringsproblem.

1.5 Avgränsningar

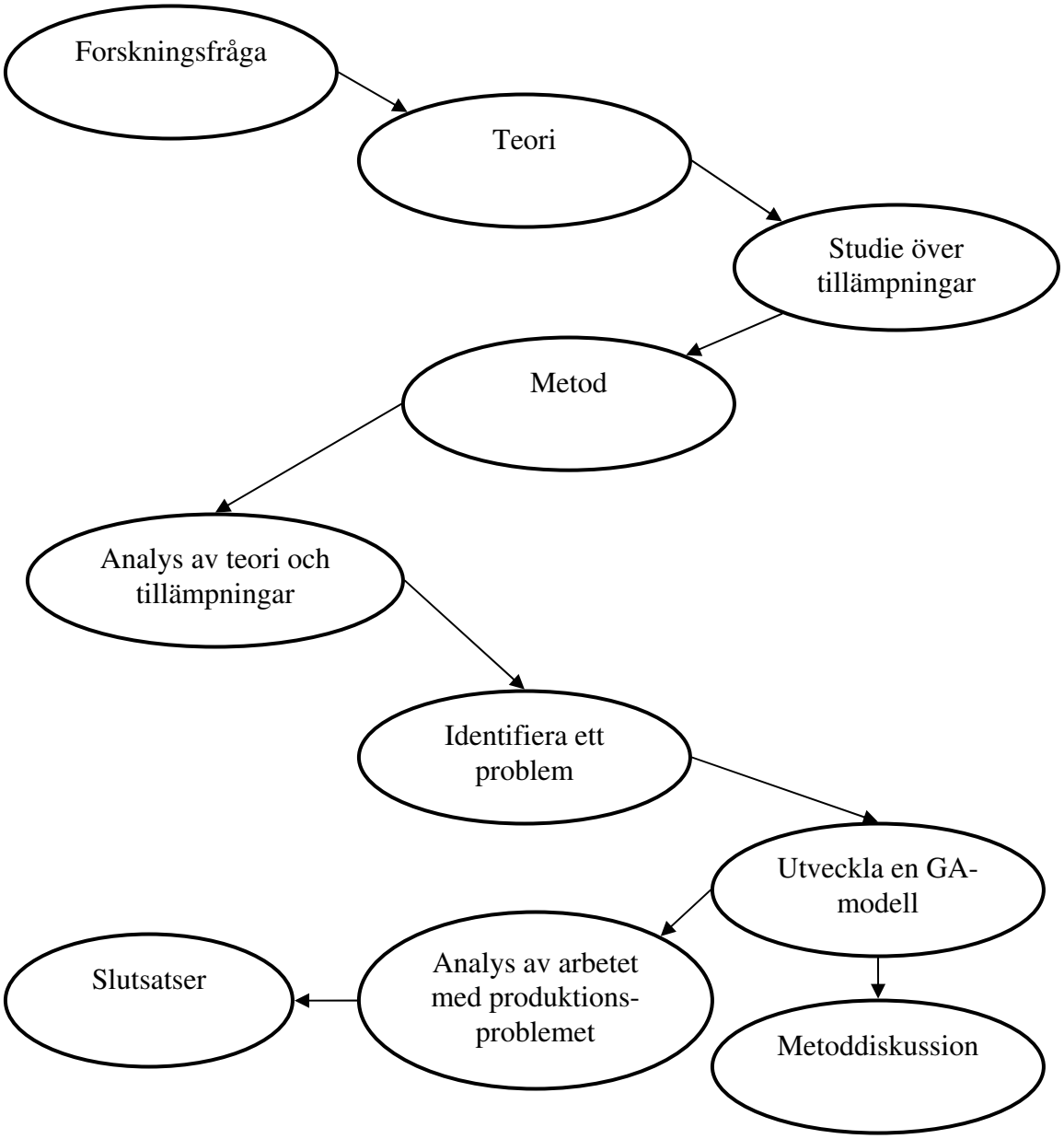
I denna uppsats avgränsar jag mig från att undersöka andra typer av artificiell intelligens eller lösningsmetoder. Jag avser endast att undersöka genetiska algoritmer, för att få en djupare förståelse över denna lösningsteknik och inte jämföra den mot andra lösningstekniker. Denna uppsats kan ses som en introduktion till genetiska algoritmer och syftet med denna uppsats är inte att utnämna någon metod som bättre än någon annan.

När det gäller min dokumentstudie över tillämpningar av genetiska algoritmer, avser jag att endast studera dokument som inriktar sig på the traveling salesman problem och hur detta problem löses med hjälp av genetiska algoritmer. Anledningen till denna avgränsning är att jag begränsar min kraft i att undersöka och förstå olika problem och istället har möjlighet att fullständigt koncentrera mitt arbete mot genetiska algoritmer och the traveling salesman problem. Valet av problem bygger på ett flertal lästa artiklar som behandlar genetiska algoritmer och bland dessa har the traveling salesman varit flitigt förekommande.

I mitt syfte nämns att jag vill utveckla en modell över en lösning där genetiska algoritmer används i ett beslutsstödjande system, för att lösa ett specifikt problem av optimerande karaktär. Jag väljer att endast angripa det specifika problemet och denna avgränsning bygger på att jag vill fördjupa mig mot det valda problemet och komma så nära en färdig prototyp som möjligt. Denna avgränsning bygger på att jag anser att det är viktigt för mig att ta del av så stora delar av utvecklingsprocessen av en genetisk algoritm, som möjligt.

Vid val av det specifika optimeringsproblemet, måste problemet vara av den karaktären att det kan vara aktuellt förekommande i ett datoriserat beslutstödjande system. Alla andra problem förkastas, då jag inte har för avsikt att undersöka problem av en annan karaktär. Anledningen till detta är min strävan att skapa en stark koppling mot beslutsstödjande system i denna uppsats.

1.6 Modell över uppsatsens upplägg



Figur 1: Modell över uppsatsens upplägg.

Figuren ovan är en modell över hur uppsatsen skall utföras. En forskningsfråga har definierats tidigare i uppsatsen (kapitel 1.3). Därefter följer ett teorikapitel i uppsatsen, som avser att ta upp nödvändig teori för att läsaren skall förstå hur genetiska algoritmer fungerar, men även förstå det sammanhang genetiska algoritmer kommer att diskuteras gentemot (se kapitel 3).

En studie över tillämningar skall sedan utföras, där tillämpningar av genetiska algoritmer skall undersökas närmare. Studien kommer helt och hållet att inrikta sig på hur genetiska algoritmer tillämpas mot the traveling salesman problem, som är ett optimeringsproblem.

Utifrån uppsatsen forskningsfråga och syfte, skall sedan en metod väljas och definieras före arbetet kan fortsätta (kapitel 3).

Därefter skall det insamlade materialet analyseras (kapitel 4), utifrån den analysmetod som presenteras i metodkapitlet. Analysen utförs i syfte att skapa en djupare förståelse för genetiska algoritmer. Därefter skall mitt arbete mynna ut i att jag identifierar ett annat problem, som genetiska algoritmer skall kunna tillämpas emot. Detta val skall bygga på mina teoretiska kunskaper och de kunskaper analysen har givit mig (kapitel 5).

I nästa steg avser jag att utveckla en modell över en genetisk algoritm, som kan användas i ett beslutsstöd, som arbetar mot det valda problemet (kapitel 5.2). Modellen över en egenutvecklad genetisk algoritm ser jag som ett sätt att öka mina kunskaper om genetiska algoritmer och stärka mina kommande slutsatser i denna uppsats. Därefter skall mitt arbeta sammanfattas, utvärderas och uppsatsens forskningsfråga skall besvaras. Detta gör jag genom att utföra en analys av arbetet med produktionsproblemet (kapitel 6), en metoddiskussion (kapitel 7) och slutligen presentera mina slutsatser (kapitel 8).

2 Teori

Följande kapitel är ett teoretiskt kapitel, som avser att behandla ämnesområden som är intressanta och nödvändiga för att förstå innehållet i uppsatsen. Av denna anledning kommer inte all teori att förekomma vid en senare analys, utan är endast med i detta kapitel i ett förklarande syfte. Problemlösning, beslutsfattande, och beslutsstödjande system kommer att behandlas. Därefter följer en presentation av genetiska algoritmer och deras olika funktioner, som exempelvis genetiska operatorer. The traveling salesman problem (TSP) kommer därefter att introduceras och förklaras för läsaren. Slutligen kommer två vetenskapliga artiklar att presenteras, som beskriver arbetet med att tillämpa genetiska algoritmer mot the traveling salesman problem. Detta kapitel har i avsikt att ge läsaren en teoretisk bakgrund och insikt i förekommande begrepp och tankesätt som existerar i uppsatsen och samtidigt beskriva hur genetiska algoritmer kan tillämpas mot ett verkligt problem.

2.1 Problemlösning och beslutsfattande

Detta kapitel försöker förklara beslutsfattande och problemlösning och de komplikationer som kan förekomma vid problemlösning eller för en beslutstagare. Detta kapitel kan ses som en introduktion för nästa kapitel, som kommer att förklara vad beslutsstödjande system är. Beslutsstödjande system har som tidigare förklarats, en övergripande roll i denna uppsats.

Lundh L., Montgomery H. & Waern Y., (1992) menar att problemlösning kan ses som en form av ett begrepp som brukar kallas för ”kontrollerad informationsbearbetning”. Detta påstående grundar sig i att problemlösning är målinriktad, där en individ målmedvetet ägnar tid och mental ansträngning åt att bearbeta en mängd information i syfte att finna en lösning på ett specifikt problem.

För människor och även andra arter från djurriket innehåller problemlösning inblandning av våra olika minnen, där korttidsminnet och långtidsminnet spelar en stor roll för oss människor. Vid problemlösning försöker vi hålla en stor mängd information i vårt minne, samtidigt som vi försöker bearbeta denna information. Om mängden information är mer än vår minneskapacitet uppstår svårigheter i problemlösningen. (Lundh L., Montgomery H. & Waern Y., 1992)

Det finns problem av olika slag och problem uppstår vid en avvikelse mellan vår inre mentala bild av något och hur vi upplever samma sak i verkligheten (Lundh L., Montgomery H. & Waern Y., 1992). Exempel på detta kan vara en individ som har kunskaper om hur ett pussel, bestående av fyra pusselbitar, ska läggas. Problemet uppstår när pusslet utökas till åtta pusselbitar och en avvikelse sker mellan den lösningsmodell individen har sedan tidigare och den nya situation som uppkommit. Pearl J. (1988) menar att många situationer går att formulera i regler av olika slag. Det finns dock alltid situationer där avvikelser från vissa regler uppkommer. Det är svårt att formulera regler

och alla undantag som kan uppkomma från regeln. Det är genom undantag en avvikelse sker och situationen övergår till problemlösning.

Beslutsfattande kan beskrivas som en process där en beslutstagare ställs inför en situation i vilket ett beslut skall fattas. Beslutsfattande kan delas in i tre huvudfaser, där varje fas måste genomgås innan ett beslut kan tas. Dessa faser är enligt följande: informationstillägnandefasen, informationsutvärderingsfasen och besluts- eller bedömningsfasen. (Lundin R., 2003)

- Informationstillägnandefasen handlar om sökmönster och de strategier som sätts i bruk för att hantera insamlingen av information. Insamling utförs i och med att en beslutstagare ställs inför en ny beslutssituation. En beslutssituation är ofta uppbyggd och beroende av ett antal olika faktorer. Information om det nuvarande tillståndet, tillgängliga resurser möjligheter och faror i beslutssituationen är därför viktig att ta vara på.

- Informationsutvärderingsfasen är den fas där den insamlade informationen bearbetas och beslutsregler tillämpas.

- Besluts- eller bedömningsfasen kommer beslutstagaren till när tillräckligt med information har utvärderats och tiden är inne att göra en bedömning över möjliga alternativ och ta ett beslut. Detta beslut grundas på informationstillägnandefasen och informationsutvärderingsfasen.

En beslutsfattare måste kunna samla in, bedöma, värdera, sammanställa och göra urval bland den information som finns inom beslutssituationen. Detta kan tydligt ses som en kognitiv process, samtidigt som den kognitiva processen påverkas av andra faktorer. Beslutsfattaren påverkas till stora delar av sina känslor och sin sociala bakgrund. Förväntningar från beslutsfattarens omgivning är exempelvis en faktor som påverkar beslutsfattandet. Detta utgör den del av socialpsykologin som påverkar beslutsfattandet. (Lundh L., Montgomery H. & Waern Y., 1992)

2.2 Beslutstödjande system

Med beslutsstödjande system, avser jag datoriserade beslutstödjande system om inget annat anges. Det typiska beslutstödjande systemet har många olika funktioner och uppgifter. Detta medför att en definition av begreppet ofta skiljer sig en hel del mellan olika författare. Beslutstödjande system består dock ofta av visa gemensamma karaktärsdrag, som snart skall beskrivas närmare. (Marakas G. 2003) Beslutstödjande system kan kort beskrivas som ett datoriserat system, som skall ge stöd i samband med beslutfattande. Systemet tar oftast inga beslut själv, utan det är användaren som arbetar med systemet som tar de slutgiltiga besluten. (Aronson J. & Turban E., 2001)

I följande stycke presenteras ett antal gemensamma karaktärsdrag för beslutstödjande system hämtat från Marakas G., (2003):

- Oftast används beslutsstödet vid semistrukturerade eller ostrukturerade problem, där beslutssituationen blir för omfattande för en mänsklig beslutsfattare.

- Beslutsstödets avsikt är inte att ersätta beslutsfattaren, utan istället ge stöd och hjälp åt en mänsklig beslutsfattare.

- Beslutsstödet bör stödja alla delar i en beslutsprocess. Det vill säga att ett komplett beslutsstöd skall stödja användaren med insamling, bearbetning, sammanställning och värdering av existerande information.

- Beslutstödet kontrolleras och styrs av en användare. Här är det tydligt att det är användaren, alltså beslutsfattaren, som styr beslutsstödet. Beslutstödet ger svar på de frågor beslutsfattaren ställt till systemet och därefter är det upp till beslutsfattaren att bestämma vad som skall hända.

- Använder bakomliggande beslutsdata och beslutsmodeller. Beslutsstödet har sedan tidigare lagrad information om vilket tillvägagångssätt som är lämpligt i en viss beslutssituation. För att kunna ge god hjälp och stöd till beslutsfattaren använder beslutstödet sig av tidigare lagrad information och modeller för att hantera beslutssituationen.

- Beslutsstödet är interaktivt och användarvänligt. Tanken med ett beslutsstöd är att en beslutsfattare med vilken bakgrund som helst, skall kunna använda sig av ett beslutsstöd. Således behövs ingen djupare datorkunskap, utan användaren kan direkt interagera med det beslutstödjande systemet.

- Beslutsstödet utvecklas ofta genom en evolutionär och iterativ process. Ett beslutsstöd genomgår ofta många förändringar och förbättringar för att ge beslutsfattaren bästa tänkbara stöd vid ett beslut.

- Ger stöd åt individuella, grupper eller teambaserade beslutsfattare.

Pearl J., (1988) menar att osäkerhet och risk är grunden till ett problem och att beslutsfattande är ett sätt att handskas med problemet och för att ta beslut om vilken åtgärd som är mest lämplig. Enligt undersökningar av Lundh L., Montgomery H. & Waern Y., (1992) kan människor ta hänsyn till ungefär fem stycken faktorer åt gången vid beslutsfattande. Det är delvis utifrån detta sista påstående som intresset för beslutsstödjande system har växt fram. Ett beslutsstöd har många fördelar, men även nackdelar. Nedan presenteras en lista över för- och nackdelar med datoriserade beslutsstödjande system hämtat från Marakas G., (2003):

<u>Fördelar</u>

- Utökar beslutsfattarens förmåga att hantera fakta och information. Enligt ovan kan människor i snitt endast hantera fem faktorer åt gången, medan en dator simultant kan hantera flera aktuella faktorer samtidigt.

- Utökar beslutsfattarens förmåga att hantera stora, tidskrävande och komplexa problem. Stora och komplexa problem blir ofta för svåra att hantera för en beslutsfattare, då kan ett beslutsstödjande system vara till hjälp.

- Korta den tid som associeras med beslutsfattande. Även om mycket tid har lags på att utveckla ett beslutsstöd, kan varje individuellt beslut få en snabbare beslutstid.

- Beslutsstödet kan öka tillförlitligheten vid beslutsfattande. Om beslutsstödet visar samma sak, som beslutsfattaren tänkte, leder detta till att beslutsfattaren kan med större säkerhet och tillförlitlighet fatta ett beslut.

- Uppmuntra en beslutsfattare att undersöka och utforska andra typer av lösningar. Människor kan ibland få ”tunnelseende” och då kan ett beslutsstöd vara en hjälp för att uppmuntra beslutsfattaren att även undersöka andra typer av lösningar, innan ett beslut fattas.

- Skapa nya bevis i en beslutssituation, som stöder en beslutsfattare, eller bekräfta viktiga faktorer för beslutssituationen.

- Skapa strategiska eller konkurrenskraftiga fördelar emot konkurrenter. Genom de många fördelar som nämnts ovan och andra icke uttalade, kan ett beslutsstödjande system ge många fördelar mot konkurrenter.

<u>Nackdelar</u>

- Beslutsstödjande system går inte att utveckla på ett sådant sätt att de innehåller mänskliga egenskaper, som ibland är viktiga vid beslutsfattande. Exempel på detta är kreativitet, föreställningsförmåga och intuition.

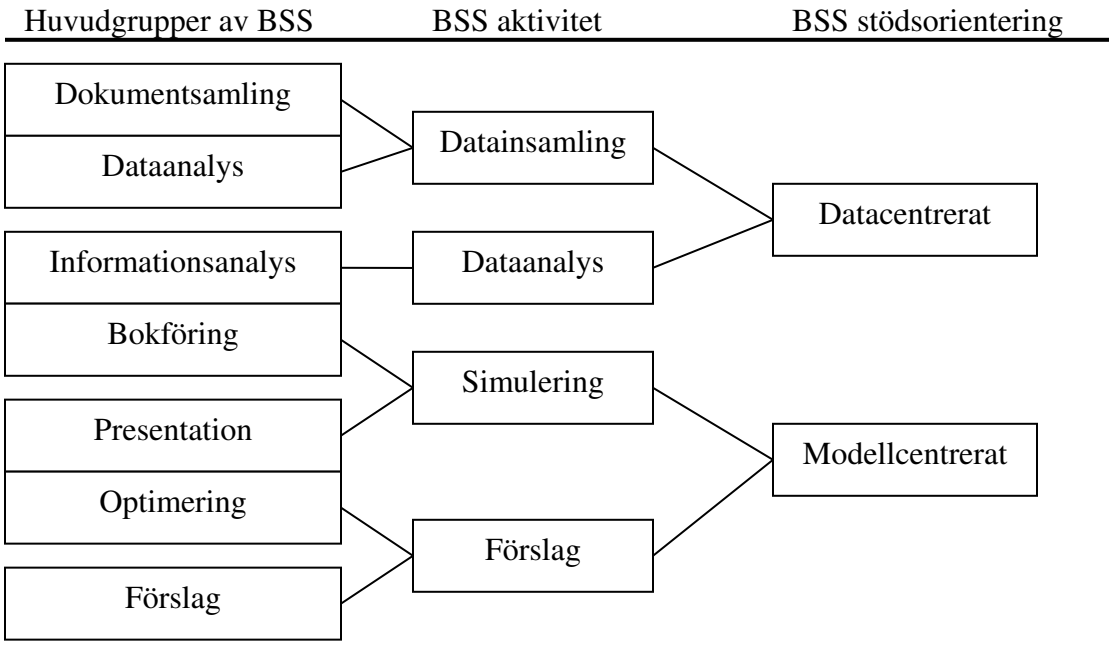
- Arbetet ett beslutsstöd kan genomföra är begränsat av det datorsystem, beslutsstödet körs på, beslutsstödets design och de kunskaper som finns i systemet.

- Språk och kommandon mellan beslutsfattare och det beslutsstödjande systemet är ännu inte tillräckligt utvecklat för att ett naturligt språk skall kunna användas.

- Beslutsstödjande system är ofta utvecklade i syfte att arbeta i ett smalt kontext. Detta stoppar beslutsstödet att vara verksam mot andra typer av problemområden.

Datoriserade beslutsstöd kan användas mer eller mindre i de flesta typer av beslutssituationer. Ekonomiska frågor, inköpsfrågor och lagerhanteringsfrågor är alla problem som naturligt uttrycks kvantitativt. Är uppgiften istället att välja bästa kandidat till VD-posten i ett företag, eftersöks vissa kvalitéer och egenskaper. För att ha möjlighet att använda ett beslutsstödjande system, vid ett beslut med kvalitativ bakgrund, omsätts ofta de kvalitativa egenskaperna till kvantitativa värden. Marakas G., (2003)

Marakas G., (2003) använder sig av en figur över klassifikationer på beslutsstödjande system. Ett beslutsstöd kan hantera olika typer av problem och många olika begrepp förekommer över benämningar av problem och problemområden. I denna uppsats använder jag mig av Marakas G., (2003) begrepp och här följer de olika klassifikationerna:



Figur 2: Klassifikation av olika beslutsstödjande system enligt Marakas G., (2003).

- *Dokumentsamling* avser system som behandlar samlingar av dokument av olika typ. Att skapa en ordning bland all lagrad information är viktigt och att underlätta för användaren att hitta den information användaren söker efter.

- *Dataanalys* avser analys av insamlad rådata. Beslutsstödet hanterar ofta insamling av data genom någon form av sensor för att sedan analysera datat.

- *Informationsanalys* är en typ av beslutsstödjande system som analyserar lagrad information. I det här fallet ligger information på en komplext högre nivå än rådata.

- *Bokföringssystem* är beslutsstöd som koncentrerar sig på ekonomiska problem och bakom bokföringssystemet ligger ekonomiska modeller för att utföra beräkningar av olika slag.

- *Presentation* är en typ av uppgift ett beslutstödjande system kan ha. Beslutsstödet skall i dessa fall presentera ett resultat på ett lämpligt sätt, som hjälper användaren att på lättöverskådligt sätt ta del av nödvändig information.

- *Optimering* är en vanligt förekommande funktion i beslutsstödjande system och går helt enkelt ut på att hitta den optimala lösningen i ett problem.

- *Förslag* är en typ av funktion flertalet beslutsstödjande system innehåller. Utifrån en given situation skall beslutsstödet generera lämpliga förslag på åtgärder och motivera för användaren varför de specifika förslagen är att föredra.

2.3 Genetiska algoritmer

Begreppet genetiska algoritmer (GA) kan låta krångligt och för att lättare ge en förståelse, delar vi upp begreppet i ”genetisk” och ”algoritm”. Den genetiska delen av begreppet handlar om evolution och ett naturligt urval enligt darwinismen, där de starkaste individerna överlever (Leou J. & Lin D., 1997). Inom biologi avses evolution vara likvärdigt med de förändringar en viss art av organismer, genomgår under ett antal generationer. Evolution är verksam på en population av en viss art, där det förekommer viss variation av egenskaper bland individerna. Evolutionen bygger på några enkla processer, nämligen: reproduktion, genetisk variation, konkurrens och urval. Dessa enkla processer leder till adaption, alltså en anpassning inom arten. (Olsson B., 1996).

Även begreppet algoritm är svårt att förstå och kan behöva en närmare förklaring. Grundtanken är att en algoritm är en noggrann plan eller metod, som stegvis beskriver genomförandet av en viss operation eller uppgift. En algoritm kan alltså ses som en lösningsmetod för ett specifikt problem (Juang C., 2004). Källkod till vanliga applikationer kan delvis ses som algoritmer för att utföra en viss operation. Exempel på detta är en växlingssfunktion för att omvandla ett givet antal svenska ören, till de olika valörer som existerar i det svenska penningsystemet.

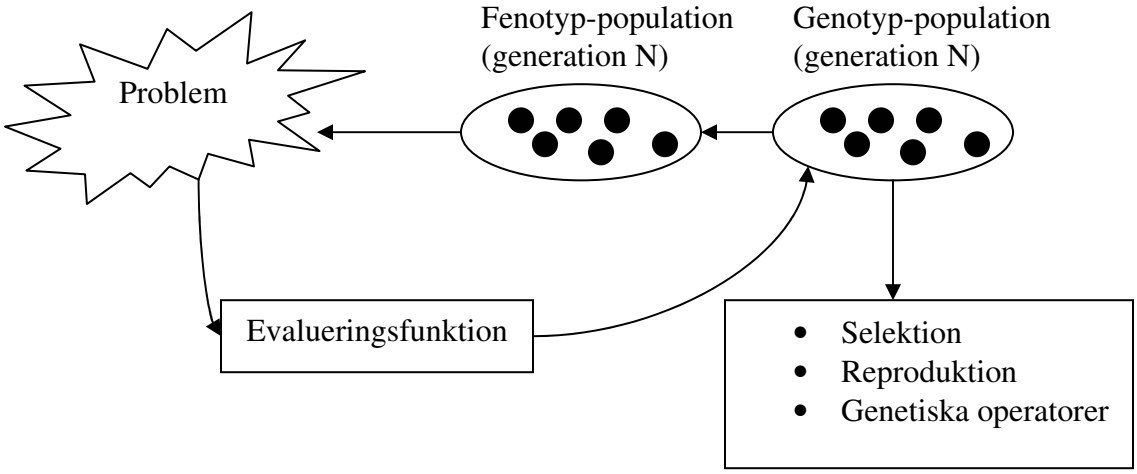
Slår vi dessa två begrepp samman bildas en metod som kallas genetiska algoritmer. Genetiska algoritmer är tänkt att på ett evolutionärt sätt förändra och förbättra en algoritm i syfte att lösa ett specifikt problem. Denna metod har inspirerats av den naturliga evolutionen vi ser i djur-, och växtriket och genetiska algoritmer försöker till så stor del som möjligt att efterlikna de egenskaper den naturliga evolutionen har. Metoden kan ses som en artificiell variant av evolution där det som förändras inte är organismers egenskaper, utan olika former av datastrukturer, exempelvis källkod. (Olsson B., 1996).

2.4 Översiktlig beskrivning av genetiska algoritmer

Genetiska algoritmer arbetar med en population av individer, även kallat kromosomer, som motsvarar ett antal algoritmer med varierande egenskaper. Dessa algoritmer är alla, mer eller mindre bra lösningar på ett givet problem. Dessa algoritmer är kodade i någon lämplig form eller uttrycka som källkod. Dessa algoritmer kallas för genotyper och populationen för en genotyp-population. (Olsson B., 1996)

Då en genotyp avkodas till ett körbart format bildas en fenotyp och en population av fenotyper kallas en fenotyp-population. En fenotyp kan direkt tillämpas på ett problem och ge ett mer eller mindre bra resultat. Vid arbete med genetiska algoritmer testas varje individ i en fenotyp-population, med hjälp av en evalueringsfunktion som ger ett sorts betyg på hur väl en specifik algoritm löser det givna problemet. Detta betyg går ofta under beteckningen fitnessvärde. (Olsson B., 1996)

Fitnessvärdet är det som ligger till grund för det fortsatta arbetet med genetiska algoritmer, nämligen urvalet eller selektionen, som ligger till grund för reproduktion. Figuren nedan försöker visa relationen mellan den genotypa-populationen, den fenotypa-populationen, de bådas individer och den evalueringsfunktion som träder i bruk, då individer från den fenotypa-populationen tillämpas. (Olsson B., 1996)



Figur 3: Översiktlig bild över genetiska algoritmer (Olsson B., 1996).

En annan beskrivning av genetiska algoritmer bygger på nedanstående sex steg, och denna beskrivning är hämtad från Leou J. & Lin D., (1997).

1. **Skapa en initial population av kromosomer (genotyper).** Den första populationen utför ofta ett helt slumpmässigt försök att lösa ett specifikt problem.

2. **Evaluera varje kromosom och tilldela den ett fitnessvärde.** Varje fenotyp testas mot problemet och får ett fitnessvärde av evalueringsfunktionen. Utifrån detta sker en selektion (urval).

3. **Skapa nya kromosomer, genom att para de utvalda kromosomerna, och applicera mutationer och crossover.** De utvalda kromosomerna från föregående generation skapar en ny generation genom reproduktion och olika genetiska operatorer.

4. **Ta bort den gamla generationen (generation N) för att ge rum för den nya generationen (generation N+1).** Gamla lösningar tar ofta bara upp onödigt med minne och kan därför tas bort. I vissa fall kan det dock finnas ett intresse att redovisa äldre lösningar, speciellt om det finns ett intresse att följa evolutionens gång för en lösning.

5. **Evaluera de nya kromosomerna och tilldela dem ett fitnessvärde.** En ny evaluering sker på samma sätt som i steg 3

6. **Om ett stoppkriterium är uppfyllt, stanna och presentera den bästa lösningen, annars återgå till steg 3.** Genom att återgå till steg 3, skapas generation efter generation, med vissa förändrade egenskaper, som förhoppningsvis kan ge en tillfredställande lösning på det specifika problemet.

2.5 Genetiska operatorer

För att få evolutionen att gå framåt och för att framställa nya generationer av individer med anpassade egenskaper används genetiska operatorer. De tre mest vanligt förekommande genetiska operatorer som forskare fokuserar på är selektion, crossover och mutation. (Hwang S. & Kuo T., 1996)

2.5.1 Evaluering och selektion

En av grundidéerna med genetiska algoritmer är ett naturligt urval, enligt darwinismen. Detta efterliknas vid användandet av genetiska algoritmer genom en evalueringsfunktion och ett urval. Det finns två olika typer av reproduktionsscheman. Dels *generationsreproduktion*, som ersätter hela den befintliga populationen med en ny. Alternativet är ett *stabil-tillståndsreproduktionsschema*, som endast ersätter ett fåtal individer i populationen med nya individer. (Hwang S. & Kuo T., 1996)

Varje individ testas med en fitnessfunktion, även kallad evalueringsfunktion och individen tilldelas ett fitnessvärde. Enkelt går det att säga att fitnessfunktionen bedömer hur lyckad varje individ är. Detta innebär att alla individer bedöms utifrån en funktion och inte efter hur väl den lyckas med sin uppgift. Fitnessfunktionens uppgift är dock att ge en rättvis bedömning för varje individ och motsvara hur väl individen skulle lyckas i den uppgift individen skall vara verksam mot i verkligheten. (Hwang S. & Kuo T., 1996)

En fitnessfunktion kan se väldigt olika ut och är alltid anpassad efter det problem som den genetiska algoritmen är tänkt att arbeta mot (Salem O. & Shahin A., 2004). Exempel på hur en fitnessfunktion kan arbeta är den fitnessfunktion som användes vid utveckling av styrprogrammet till en robothund som skulle lära sig att gå. Till hundens fysiska kropp kopplades en vanlig datormus, som fick agera en sorts sensor i syfte att mäta hur långt robothunden hade förflyttat sig. Datormusen fick fungera som ett mätinstrument och den uppmätta sträckan blev det fitnessvärde som den enskilda individen blev tilldelad i evalueringsprocessen. (Nordin P. & Wilde J., 2003)

En lämplig fitnessfunktion i en situation då the traveling salesman problem (se kapitel 2.6) skall lösas, är att beräkna den totalt resta sträckan och tilldela de individer med kortast ressträcka, ett högre fitnessvärde. Fitnessfunktioner ska alltså utvecklas och anpassas till specifika problem. (Li G. & Louis S., 2000)

När ett fitnessvärde tilldelats alla individer i en population skall en selektionsprocess påbörjas som baseras på individernas fitnessvärden. Selektionsmetoder spelar en stor roll vid arbete med genetiska algoritmerna i syfte att hitta bättre lösningar och även skapa populationer, som innehåller en viss variation för att populationen ska kunna utveckla sig. Enligt neo-darwinism finns det tre olika grupper av urval: stabiliserande urval, ledande urval och splittrande urval. Vid användandet av genetiska algoritmer försöker dessa tre grupper av urvalsmetoder att efterliknas på olika sätt. Nedan följer en kort beskrivning av de tre grupperna av urval: (Hwang S. & Kuo T., 1996)

- *Stabiliserande urval*, även kallat normalt urval, arbetar på ett sådant sätt att individer med extrema egenskaper förkastas.

- *Ledande urval* försöker höja eller sänka medelvärdet hos populationen.

- *Splittrande urval* eliminerar individer med måttliga värden och på detta sätt skapa en population med kraftig variation.

Olsson B., (1996) beskriver och exemplifierar något han kallar för fitness-proportionell selektion. Detta innebär att urvalet eller selektionen skall ske proportionellt mot det fitnessvärden individerna har, det vill säga att individer med högre fitnessvärde har större chans att ingå i urvalet. Detta typ av urval beskrivs närmare i följande text.

För att urvalet skall kunna ske proportionellt beräknas hur stor andel varje individs fitness utgör av hela populationens summerade fitness. Varje individs chans att ingå i urvalet får på så sätt ett procentuellt värde. Urvalsprocess kan på så sätt liknas vid att snurra ett roulettehjul där varje individ är ett fält på roulettehjulet och fälten är proportionellt stora i förhållande till individens fitnessvärde. (Olsson B., 1996)

För att exemplifiera evalueringsprocessen och urval enligt ett fitness-proportionellt synsätt följer ett exempel hämtat från Olsson B., (1996):

Nr	Kromosom	Fitness
1:	1111000111	7
2:	1110110101	7
3:	1001110010	5
4:	0110110000	4
5:	0011010110	5
6:	0100000010	2
7:	1011010010	5
8:	0101111000	5
9:	1001001010	4
10:	1011000110	5
Generation		= 0
Medel-fitness		= 4.9
Högsta fitness		= 7

Figur 4: Population bestående av tio individer och deras fitnessvärde (Olsson B., 1996).

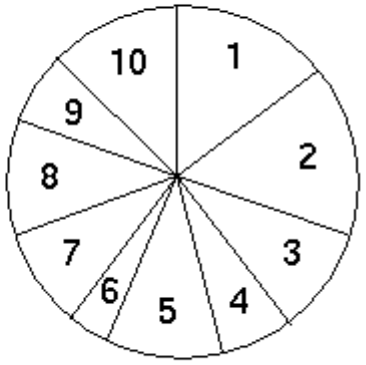
En individ eller kromosom är i detta exempel en sträng som innehåller ett antal nollor och ettor. Det som eftersträvas är en sträng med så många ettor som möjligt och en fitnessfunktion för detta beräknas antalet ettor i strängen och fitnessvärde för en individ är alltså antalet ettor i strängen (lägst 0 och högst 10).

Nr	Kromosom	Fitness	Andel
1:	1111000111	7	0.143
2:	1110110101	7	0.143
3:	1001110010	5	0.102
4:	0110110000	4	0.082
5:	0011010110	5	0.102
6:	0100000010	2	0.041
7:	1011010010	5	0.102
8:	0101111000	5	0.102
9:	1001001010	4	0.082
10:	1011000110	5	0.102
S:a		49	1.000

Figur 5: Population bestående av tio individer med varje individs andel av populationens totala fitnessvärde, som skall ligga till grund för fitness-proportionell selektion (Olsson B., 1996).

Enligt det fitness-proportionella synsättet skall varje individs andel av den totala fitnessen beräknas. Totala fitnessen för populationen beräknas till 49 och individ nummer 1 har fitness 7. Detta leder till 7 / 49 = 0,143. På samma sätt beräknas andelen för varje individ i populationen.

Nr	Kromosom	Fitness	Andel
1:	1111000111	7	0.143
2:	1110110101	7	0.143
3:	1001110010	5	0.102
4:	0110110000	4	0.082
5:	0011010110	5	0.102
6:	0100000010	2	0.041
7:	1011010010	5	0.102
8:	0101111000	5	0.102
9:	1001001010	4	0.082
10:	1011000110	5	0.102
	S:a. 49		1.000



Figur 6: Andelen av varje individs fitnessvärde i form av ett roulettehjul (Olsson B., 1996).

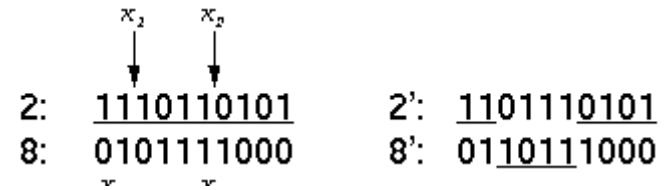
För att selektionen skall bli fitness-proportionell har alltså individ nummer 1 och 2 större chans (14,3 procent) att ingår i selektionen än exempelvis individ 7 och 8 som har en 10,2 procent chans. Detta motsvaras i bilden ovan av ett roulettehjul där individ nummer 1 och 2 har fått större fält än individ nummer 7 och 8. Ett slumpmässigt urval kan ske efter dessa förutsättningar för att uppnå en fitness-proportionell selektion.

Snurrar vi på hjulet ovan sex gånger får vi exempelvis ett utfall på nummer 9, 10, 2, 8, 5 och 7. Lägg märket till att det inte är säkert att de individerna med högst fitness kommer att ingå i urvalet. I längden kommer dock urvalet av individer att ske direkt proportionellt mot individernas fitnessvärde. Dessa sex individer som blivit utvalda enligt ett fitness-proportionell synsätt, bildar nu 3 par av föräldrar till en ny generation av individer. (Olsson B., 1996)

2.5.2 Crossover

Crossover är en funktion inom genetiska algoritmer som enligt många forskare är väldigt viktig. Det finns även forskare som menar att genetiska algoritmer utan crossover, inte längre skulle vara genetiska algoritmer. Crossover som funktion i genetiska algoritmer baseras på samspelet i naturen, där två individer parar sig och ett utbyte av kromosomer sker och det bildas avkomma. (Salem O. & Shahin A., 2004)

Salem O. & Shahin A. (2004) och Olsson B. (1996) nämner båda två-punkts crossover för att exemplifiera hur crossover kan fungera. Till varje par av individer tilldelas två slumpvisa överkorsningspunkter. Den information som är mellan dessa två överkorsningspunkter i de två individerna, byter helt enkelt plats. Individ nummer ett får på så sätt ta del av den andra individen, medan individ nummer två ger en del till individ nummer ett, samtidigt som den får ta del av ny information, som kommer från individ nummer ett.



Figur 7: Överkorsning mellan två individer genom två-punkts crossover (Olsson B., 1996).

I exemplet ovan har individerna från två och åtta valt ut som föräldrar och en överkorsning skall ske. Enligt två-punkts crossover slumpas två överkorsningspunkter, dessa är i exemplet mellan andra och tredje tecknet i strängen och mellan sjätte och sjunde tecknet. Det som finns mellan dessa överkorsningspunkter byter helt enkelt plats mellan de två individerna och på så sätt bildas två förändrade individer genom överkorsning. (Salem O. & Shahin A., 2004) De två nya individerna syns till höger i bilden ovan. De värden som är understrukna härstammar från individ nummer två och de värden som inte är understrukna härstammar från individ nummer åtta.

Två-punkts crossover är en enkel form av crossover mellan individer, men går att utveckla och använda sig av olika restriktioner för att uppnå bättre resultat (Salem O. & Shahin A., 2004). Aritmetisk crossover (Leou J. & Lin D., 1997) och likformig crossover (Legault G. & Pierre S., 1998) är två andra typer av crossover.

2.5.3 Mutationer

Mutation är en funktion hos genetiska algoritmer som slumpvis förändrar en del hos en individ. Mutation har ofta en sekundär roll efter crossover och används bara ibland. Det är vanligt förekommande att användaren får bestämma hur frekvent mutationer skall uppkomma före den genetiska algoritmen startas och börjar arbeta. (Legault G. & Pierre S., 1998)

Samma författare som ovan exemplifierar även detta med att säga att en användare bestämmer att sannolikheten för att en mutation skall ske är 0,005. Detta innebär att det är 5 chanser på tusen att en mutation skall genomföras. Då en mutation sker slumpas en position hos en individ och det värdet på denna position förändras på något sätt, med vissa restriktioner. (Legault G. & Pierre S., 1998)

I verkligheten och i naturen är sannolikheten att mutationer uppkommer relativt liten. I genetiska algoritmer bestäms denna sannolikhet genom någon regel och en grundregel är att om individens består av en längre sträng så ska sannolikhetsgraden, för att mutationer ska uppkomma, hållas lägre. Detta på grund av att varje del av individen testas om en mutation skall uppkomma och består individen av en längre sträng, leder det till att den totala sannolikheten för att en mutation skall uppkomma ökar totalt för individen. (Salem O. & Shahin A., 2004)

En viktig egenskap som mutationer medför är att återföra egenskaper som tidigare har försvunnit genom crossover eller redan i ett urval. Det är inte alltid viktiga egenskaper lever vidare, men med mutationer får dessa egenskaper en chans att återigen komma tillbaka till populationen. (Legault G. & Pierre S., 1998) Genom mutation förstoras omfånget av den aktuella lösningen och lösningen får även en möjlighet att undvika lokala maximipunkter (Leou J. & Lin D., 1997).

2.5.4 Inversion

Inversion är en annan typ av genetisk operator, som färre författare nämner i litteraturen. Legault G. & Pierre S., (1998) presenterar dock denna operator, vars funktion är att vända på en individ. Värdet på position ett byter plats med position L (L motsvarar längden på individen) och värdet på position två byter plats med L-1 (alltså till den näst sista positionen i individen).

Även detta är en metod för att skapa förändringar och variation mellan individer. Det är även en operator som har ett snarlikt syfte som mutationer. Där egenskaper hos individer som tidigare har försvunnit genom selektion, än en gång har en chans att komma tillbaka till populationen. Detta medför som nämnts tidigare, att omfånget av den aktuella lösningen förstoras och lösningen får även en möjlighet att undvika lokala maximipunkter (Leou J. & Lin D., 1997).

2.6 The traveling salesman problem (TSP)

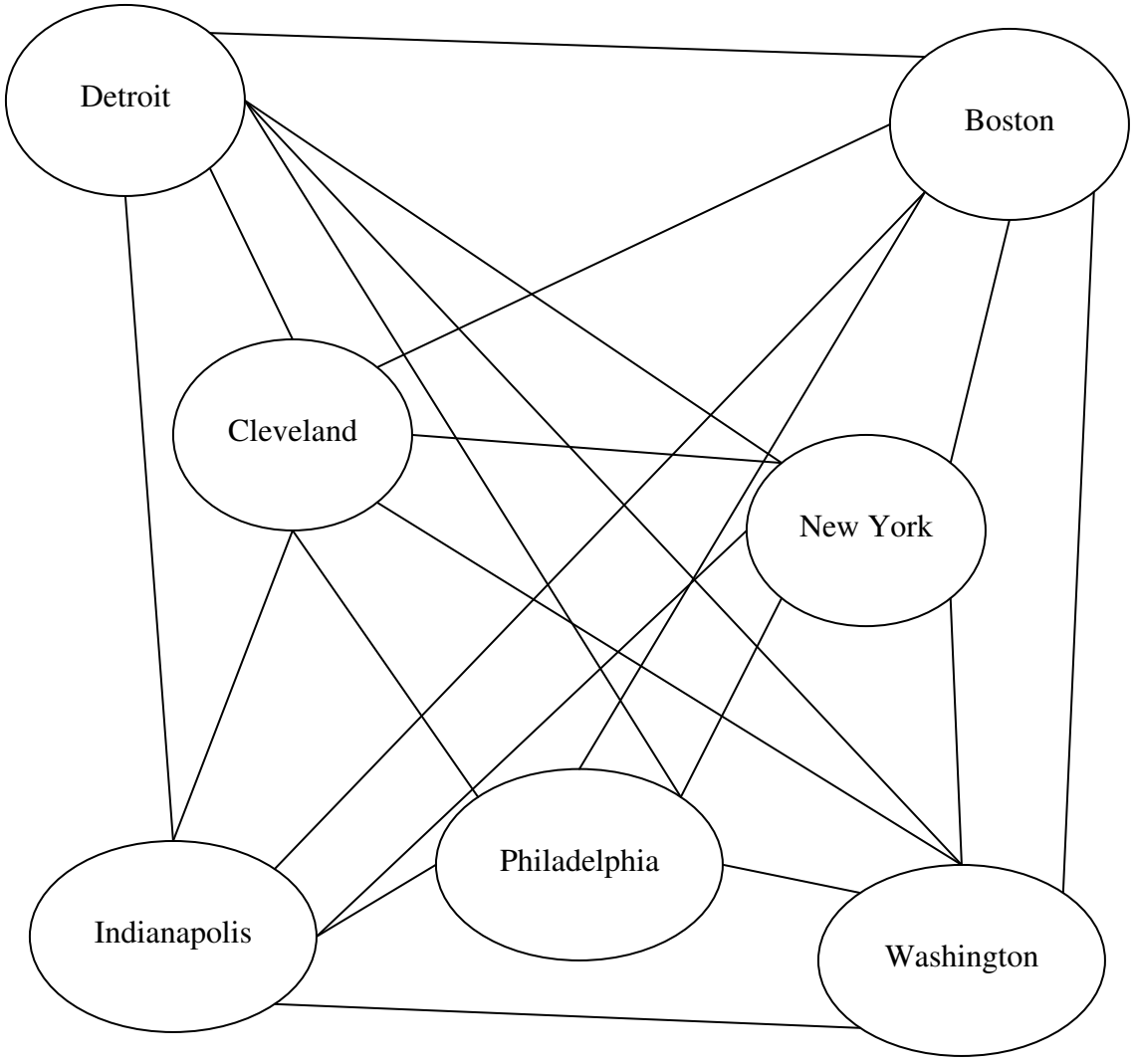
Problemet med den resande försäljaren eller the traveling salesman problem (TSP) är ett känt optimeringsproblem som under årtionden har fascinerat forskare och entusiaster. En av de tidigaste publicerade artiklar över TSP är *”On the Hamiltonian Game (a traveling-salesman problem)”*, skriven av J. B. Robinson (1949). (Harmel B. & Patterson M., 2003).

TSP bygger på att hitta den rutt som minimerar den totalt resta sträckan. Kraven på rutten är att den skall starta i en stad och därefter skall ett antal (n) städer vara besökta en gång och endast en gång och att rutten slutligen skall avslutas i den stad rutten påbörjades. (Tsai H., Yang J., Tsai Y. & Kao C., 2004) Det är vanligt att presentera TSP som en tabell och nedan följer en tabell och ett diagram som försöker visa komplexiteten i TSP (Harmel B. & Patterson M., 2003).

	Till:	**Boston**	**Cleveland**	**Detroit**	**Indianapolis**	**New York**	**Philadelphia**	**Washington D.C.**
Från:	**Boston**	**0**	**628**	**695**	**906**	**206**	**296**	**429**
	Cleveland	**628**	**0**	**170**	**294**	**473**	**413**	**346**
	Detroit	**695**	**170**	**0**	**278**	**637**	**576**	**506**
	Indianapolis	**906**	**294**	**278**	**0**	**713**	**633**	**558**

New York	**206**	**473**	**637**	**713**	**0**	**100**	**233**
Philadelphia	**296**	**413**	**576**	**633**	**100**	**0**	**133**
Washington D.C	**429**	**346**	**506**	**558**	**233**	**133**	**0**

Figur 8: Sträckan (i miles) mellan städerna i ett TSP (Harmel B. & Patterson M., 2003).



Figur 9: Schematisk bild över ett TSP (Harmel B. & Patterson M., 2003).

Den schematiska bilden över TSP visar på de olika alternativ den resande står inför i varje stad. Diagrammet ovan försöker visa på den komplexitet en resande står inför vid varje stad, då inte alltid en resa till den närmaste staden ger den kortaste totala rutt. Bilden ovan exemplifierar antalet möjliga ressträckor mellan sju olika städer eller noder.

Då antalet städer ökar, ökar samtidigt antalet ressträckor och problemet blir allt svårare att hantera.

Det som gör TSP intressant är att det inte bara går att tillämpa på planering av rutter, utan även är tillämpbart inom andra områden. Vilket gör att TSP har blivit ett klassiskt optimeringsproblem, med ett extremt stort sökområde vilket gör problemet mycket svårt att lösa. Forskare har provat att använda sig både av exakta och heuristiska metoder för att angripa TSP. Exakta metoder lämpar sig bra i mindre problem, medan heuristiska metoder som exempelvis simulated annealing[3], är nödvändigt för att lösa större problem. (Li G. & Louis S., 2000).

2.7 Presentation av tillämningar

I följande del av denna uppsats skall två artiklar presenteras i form av en dokumentstudie. Dessa två artiklar har med olika ansatser angripit the traveling salesman problem, med hjälp av genetiska algoritmer. Dessa två artiklar har valts ut bland ett stort antal vetenskapliga artiklar och urvalet bygger på att jag som författare tycker att dessa två artiklar dels beskriver hur genetiska algoritmer verkligen tillämpas mot ett optimeringsproblem, samtidigt som artiklarna har två vitt skilda ansatser till att lösa det aktuella problemet. På detta sätt kan jag presentera två artiklar som visar en viss bredd över hur genetiska algoritmer kan tillämpas mot TSP. Bakgrund, teori, arbetssätt och författarnas slutsatser skall presenteras. I kapitel fyra följer en analys av denna uppsats teoretiska kapitel och dokumentstudie, enligt det tillvägagångssätt som beskrivs i uppsatsens metodkapitel.

2.7.1 Experiment över genetiska operatorers inverkan för framgången att lösa TSP

Hela följande kapitel är en dokumentstudie och refererar till Tsai H., Yang J., Tsai Y. & Kao C. (2004). Detta är en dokumentstudie som undersöker utförd forskning och experiment över genetiska operatorers inverkan för framgång att lösa TSP.

Artikeln presenterar TSP som ett välkänt optimeringsproblem och berättar att det finns många olika angreppssätt för att hantera problemet. I artikeln angrips TSP med hjälp av genetiska algoritmer, eller evolutionära algoritmer som författarna kallar det. Utifrån kända styrkor och svagheter i vanliga genetiska operatorer, vill författarna med ett experimentellt tillvägagångssätt undersöka och visa på att en kombination av två genetiska operatorer kan kompensera varandras svagheter. Genom familjekonkurrens och heterogen parselektion skall den genetiska algoritmen kunna upprätthålla en varians inom

[3] Simulated annealing är en metod som används för att hitta optimala numeriska värden. Den väljer nästa steg och vid optimering är alla uppåtgående steg bra, men även vissa nedåtgående steg kan även accepteras. Detta beror på att ”dåliga” steg, ibland kan vara nödvändiga för att undvika lokala max punkter. Golden B., Pepper J. & Wasil E. (2002)

populationen. Detta har visat sig vara en väldigt viktig egenskap, för att kunna hantera och lösa TSP.

Sedan tidigare har många försökt närma sig TSP och lösa det genom att använda sig av genetiska algoritmer och försök har utförts i syfte att förbättra de genetiska algoritmerna. Att designa TSP-specifika genetiska operatorer är vanligt och bland dessa finns lokala sökmetoder, positionsbaserade operatorer, intervallsbaserade operatorer och kantbaserade operatorer. Alla dessa operatorer arbetar på olika sätt, antingen för att upprätthålla en varians inom populationen eller för att ett möjliggöra ett arv av funna dellösningar, som klassas som goda. Metoder som har integrerade domänspecifika kunskaper, kallas för memetiska algoritmer.

Baserat på artikelns analys om existerande försök att förbättra effektiviteten i användningen av genetiska algoritmer mot TSP, vill författarna utveckla ett nytt angreppssätt. Efter detta utvecklades en heterogen selektiv evolutionär algoritm, som förkortas HeSEA. Algoritmen har utvecklats utifrån edge assembly crossover (EAX) och Lin-Kernighan local search (LK), som visat sig vara två starka genetiska operatorer och beskrivs närmare i artiklen. Den framtagna HeSEA har sedan testats mot 16 välkända TSPs där antalet städer varierar från 318 till 13509.

I artikeln presenteras forskning som undersöker fyra olika kantbaserade genetiska operatorer, som är specifika för TSP. Dessa fyra kantbaserade operatorerna testades sedan mot åtta stycken enklare testproblem. Kantbaserade operatorer har i uppgift att behålla funna kanter i problemet. Dessa fyra operatorer testades mot samma åtta problem, utan att blanda in några andra modifieringar. Undersökningen visade att EAX gav det bästa resultatet.

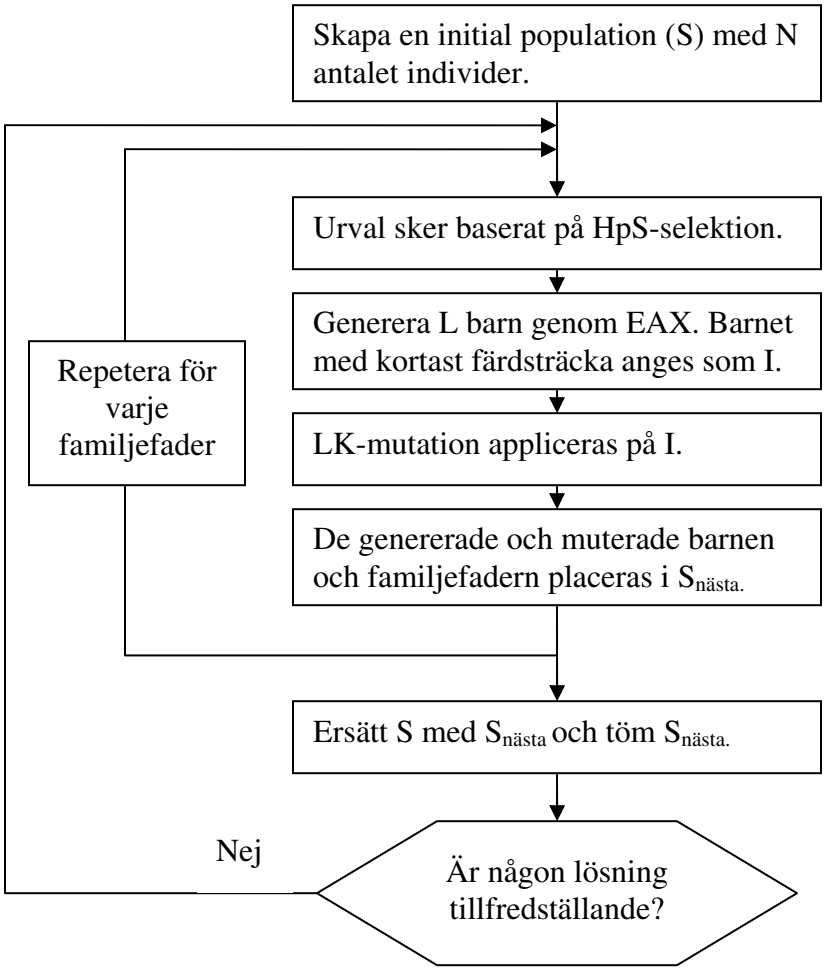
En utvärdering av dessa resultat har utförts och slutsatsen är att en bra crossoveroperator till TSP, skall vara kantbaserad och vara bra på att skapa och behålla starka kanter. EAX visade sig vara den bästa av de fyra testade operatorerna och därför valde författarna att använda sig av EAX vid utvecklingen av HeSEA.

Ett liknande angreppssätt användes för att testa LK, NJ, 2-opt och 3-swap, som är av en annan typ av genetiska operatorer. De fyra operatorerna testades mot åtta testproblem och resultaten visade på att LK den starkaste operatorn. Detta resultat ledde till att författarna valde att använda sig av LK-operatorn i den fortsatta forskningen. Den gemensamma anledningen till att EAX och LK har presterat bra i den utförda undersökningen är deras förmåga att tidigt skapa bra kanter i lösningen av ett TSP, samtidigt som den genom tiden bibehåller de goda kanterna.

EAX och LK presterar båda bra i mindre TSP, men om ett TSP innehåller fler än 4000 städer, behöver EAX väldigt stora populationer, vilket är tidskrävande. LK hittar snabbt lokala optimala lösningar, men kan ofta inte förbättra lösningen. Utifrån dessa resultat och slutsatser, har författarna valt att skapa en förbättrad genetisk algoritm, som kombinerar EAX och LK. Dessa två operatorenas fördelar kombineras och operatorerna kompenserar varandras nackdelar.

HeSEA blir namnet på den nya heterogena, selektiva evolutionära algoritm som forskarna tagit fram. Figuren nedan är en modell över hur HeSEA arbetar. Först skapas en initial population (S). Efter att varje individ genomgått evalueringsfunktionen väljs en familjefader. Familjefadern applicerar därefter heterogen parselektion (HpS) på sig själv och en annan individ från föregående population. Genom en EAX-operator genereras L barn, där barnet med kortast färdsträcka anges som I.

Därefter muteras I genom en LK-operator, för att bibehålla variation inom populationen. De genererade och muterade barnen och familjefadern placeras i $S_{nästa}$, som ersätter den föregående populationen S och $S_{nästa}$ töms för att lämna plats åt en kommande population. Bästa lösningen testas, för att bestämma om lösningen är tillfredställande, annars upprepas samma procedur mot den nya populationen S.



Figur 10: Schematisk bild över HeSEA (Tsai H., Yang J., Tsai Y. & Kao C., 2004).

Den nya framtagna genetiska algoritmen HeSEA har sedan testats experimentellt mot sexton TSP-problem. Alla problemen var hämtade från TSPLIB, vilket är en samling TSP-problem som författarna använt sig av. Antalet städer varierar mellan 318 till 13509 stycken, i de olika problemen.

Problem	Genomsnittlig tid angivet i sekunder	Bästa lösning (fel %)	Genomsnittlig lösning (fel %)	Antal optimala lösningar
Lin318 (42029)	12.4	42029 (-)	42029 (-)	20
pcb442 (50778)	9.2	50778 (-)	50778 (-)	20
Att532 (27686)	15.3	27686 (-)	27686 (-)	20
U574 (36905)	23.6	36905 (-)	36905 (-)	20
rat786 (8806)	39.1	8806 (-)	8806 (-)	20
pr1002 (259045)	91	259045 (-)	259045 (-)	20
Vm1084 (239297)	80.6	239297 (-)	239297 (-)	20
pcb1173 (56892)	84.5	56892 (-)	56892 (-)	20
u1432 (152970)	107	152970 (-)	152970 (-)	20
Vm1748 (336556)	141	336556 (-)	336556 (-)	20
u2152 (64253)	211	64253 (-)	64253 (-)	20
pr2392 (378032)	208	378032 (-)	378032 (-)	20
Pcb3038 (137694)	612	137694 (-)	137694 (-)	20
Fnl4461 (182566)	2349	182566 (-)	182566.9 (0.0005)	16
frl5915 (565530)	2773	565530 (-)	565530.5 (0.0001)	19
usa13509 (19982859)	34984	19983361 (0.0025)	19984334 (0.0074)	0

Figur 11: Tabell över experiment utförda med HeSEA mot sexton stycken TSP (Tsai H., Yang J., Tsai Y. & Kao C., 2004).

Tabellen ovan är visar de resultat de utförda experimenten givit. HeSEA testades experimentellt mot de sexton TSP-problemen, där varje test utfördes tjugo gånger. Experimenten genomfördes på en Pentium 4 1.3GHz PC, med 1GB RAM-minne. Prestandan på datorn experimentet genomfördes på inverkar givetvis på resultatet.

Tabellen ovan visar att HeSEA hittade den optimala lösningen till alla TSP-problem, förutom usa12509, som är det problemet med flest antal noder. HeSEA hittade vid varje försök den optimala lösningen till problemen, som hade färre än 4000 noder. Den bästa lösningen HeSEA hittade till usa12509, är endast 0,0025 procent över den optimala lösningen. Till de två problemen som översteg fyratusen noder, förutom usa12509-problemet, hittade HeSEA den optimala lösningen sexton respektive nitton gånger. Den genomsnittliga differensen från de optimala lösningarna var endast 0,0005 procent och 0,0001 procent.

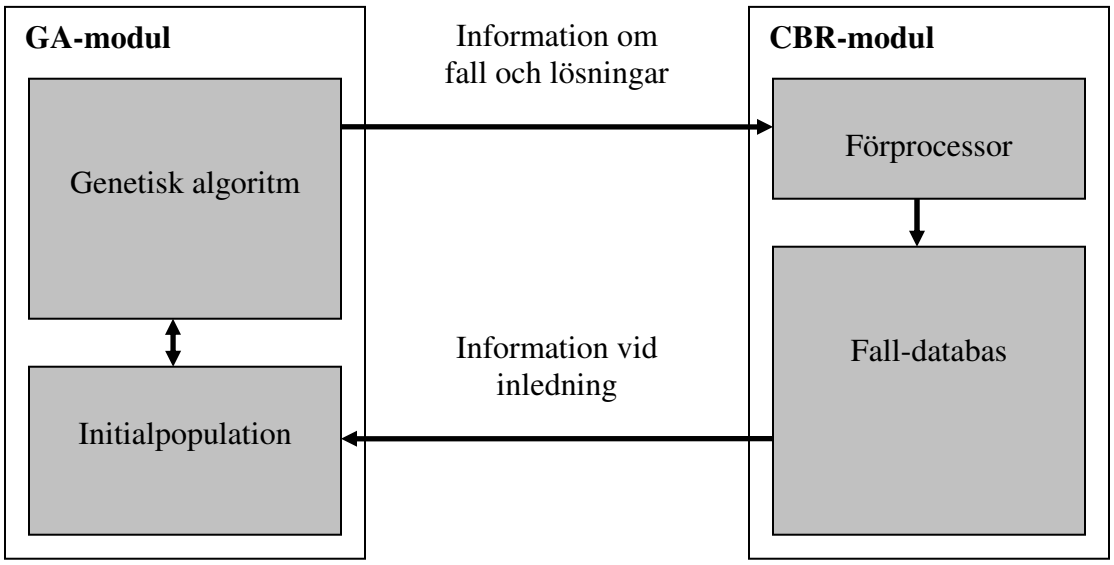
2.7.2 Att förbättra genetiska algoritmer genom att dra nytta av tidigare kunskaper

Hela följande kapitel är en dokumentstudie och refererar till Li G. & Louis S. (2000). Detta är en dokumentstudie som undersöker och utför en studie över hur genetiska algoritmer kan förbättras genom att dra nytta av tidigare erfarenheter. Tidigare fall av TSP-problem, där en genetisk algoritm har skapat en bra lösning, skall kunna användas för att förbättra och effektivisera den genetiska algoritmen.

Artikeln beskriver hur forskarna undersöker möjligheten att använda sig av genetiska algoritmer, förstärkt med ett långtidsminne för att kunna angripa likartade TSP. Tanken är att undersöka om en genetisk algoritm kombinerat med ett inlärningssystem och en falldatabas kan ge bättre resultat än en ensam genetisk algoritm. Istället för att börja med en slumpmässig initial population, kan initial populationen skapas utifrån systems tidigare erfarenheter och på så sätt få en prestandahöjning.

Inlärning kräver ett minne där erfarenheter kan lagras. Grundformen av genetiska algoritmer saknar möjlighet till inlärning och då även ett minne. Detta leder till att genetiska algoritmer i sin grundform, alltid tvingas börja från början. Det finns inget direkt förnuft i att varje gång börja om från början, men det är så många genetiska algoritmer arbetar. För närvarande finns det försök att skapa genetiska algoritmer som använder sig av maskininlärning, där regler skapas för att lagra tidigare erfarenheter. Ansatsen och tankesättet som genomsyrar denna forskning, bygger på att det finns problemområden som är mer lämpade för fallbaserad lagring.

Forskningen bygger på idéer från case-based reasoning (CBR) som handlar om att information om tidigare problem och lösningar lagras i en falldatabas, för att senare kunna användas som hjälp i nya fall. Figur 11 visar en konceptuell modell över hur ett system där en genetisk algoritm arbetar tillsammans med en typ av CBR-system.



Figur 12: Konceptuell modell över genetiska algoritmer i samarbete med en CBR-modul Li G. & Louis S. (2000).

En genetisk algoritm är inriktad på adaption och genererar lösningar. Detta liknas med en sökning, som undersöker en del av en enorm sökrymd. Varje lösning testas mot en fitnessfunktion och lösningen tilldelas ett fitnessvärde. Information om lösningar skickas till CRB-modulen, som tar vara på information om problemet och de genererade lösningarna. Vid vanlig användning av genetiska algoritmer förkastas alla lösningar, när

programmet avslutas. Med hjälp av CBR-modulen skall information om specifika fall (problem och lösningar) lagras i CBR-modulens falldatabas.

Författarna vill även poängtera att de vill undersöka om ett system kan lära sig och på så sätt förbättra en genetisk algoritm. TSP används endast som ett exempel problem och att inga speciella förbättringar genomförs på den genetiska algoritmen för att anpassa den mot TSP för att kunna generera bättre lösningar. En teoretisk modell över genetiska algoritmer måste dock anpassas för att ens kunna användas på TSP. En lämplig fitnessfunktion definierades. Presentationen av individer definierades på ett sådant sätt att färdsträckan för varje lösning visade på ordningen de olika städerna besöks. Om exempelvis det finns tre städer och stad nummer 1 besökts först, därefter 3 och därefter 2, presenterades individen som [1, 3, 2].

Traditionella crossoveroperatorer är inte heller användbara för TSP. Otillåtna barn skulle nämligen genereras genom en traditionell crossoveroperator, då exempelvis samma stad kan förekomma mer än en gång och vissa städer kan saknas helt i en individ. Forskarna använder sig av greedy crossover, som väljer den första staden i en förälder, jämför den närmaste staden i både föräldrarna och väljer därefter den som ligger närmast. Om en stad redan finns representerad i individen, väljs den näst närmaste staden och om även den finns väljs en slumpmässig stad, bland de som ännu inte finns representerade.

Även mutationer kan skapa otillåtna barn och traditionella mutationsoperator är därför inte användbara. Mutation fungerar istället på det sättet att operatorn slumpmässigt väljer ut två bitar från en individ och byter plats på dem. På detta sätt skapas en mutation, men individen representerar fortfarande en tillåten lösning. Selektionen utförs enligt rouletthjulsprincipen och den bästa individen går alltid vidare till nästa generation.

Även en CBR-modul utvecklades och kopplades mot GA-modulen. För att experimentellt testa hur denna kombination presterade mot TSP-problem utvecklades en metod för att genomföra experimentet. Fyra stycken TSP-problem fastslogs och presenteras i figur 12. Experimentet skulle utföras genom att GA-modulen tillsammans med CBR-modulen fick arbete mot de fyra TSP-problemen. När detta var utfört och falldatabasen byggts på med information, modifierades de fyra TSP-problemen enligt figur 13.

Antal städer	Populationsstorlek	Antal generationer
52	200	300
76	250	400
105	300	400
127	300	500

Figur 13: Matris över de fyra TSP-problemen, den populationsstorlek som användes och antalet generationer den genetiska algoritmen kördes (Li G. & Louis S., 2000).

Problem (typ av förändring)	Storlek
Samma	Samma
Ändra en nod	Samma

Ändra två noder	Samma
Lägg till en nod	En mer
Ta bort en nod	En mindre

Figur 14: Matris över hur de ursprungliga TSP-problemen modifierades (Li G. & Louis S., 2000).

Experimentet fortlöpte genom att de fyra omodifierade TSP-problem angrepps med hjälp av en enkel genetisk algoritm, som beskrivits ovan. Information om problemet och de genererade lösningarna sparades i CBR-modulen. Efter detta genomfördes experiment mot modifierade TSP-problem (enligt figur 13). Initialpopulationen skapades utifrån information från CBR-modulen och resultatet av experimentet visas nedan i figur 14.

Problem	Första generationen	Sista generationen
Slumpmässigt	491,64%	111,19%
Samma problem	109,58%	104,97%
Ändra en nod	112,55%	106,49%
Ändra två noder	119,39%	106,05%
Lägg till en nod	109,48%	105,36%
Ta bort en nod	116,49%	107,65%

Figur 15: Genomsnittlig sträcka per problem, beräknat utifrån den optimala lösningen (Li G. & Louis S., 2000).

Matrisen ovan presenterar resultatet av det utförda experimentet, där GA och CBR kombinerats, för att lösa TSP-problem med vissa modifieringar. Vid slumpmässiga TSP-problem gav första generationen en genomsnittlig sträcka på 491,64 procent av den optimala sträckan, alltså nästan fem gånger längre än den optimala sträckan. Användes istället CBR mot ett problem som systemet tidigare arbetat mot, genererades en initial population, som genomsnittligt låg på 109,58 procent av den optimala sträckan. Detta innebär att den första generationen genomsnittligt endast var ~10 procent längre än den optimala sträckan. Vid den sista generationen hade lösningen även förbättrats och lösningen var endast ~5 procent längre än den optimala sträckan för problemet. Liknande resultat syns på alla de olika modifieringarna.

Resultaten av experimentet visar på att injektion av lösningar av liknande problem, alltid ger bättre lösningar än att börja med en slumpmässig initialpopulation. Detta bevisar att addera en CBR-modul, som komplement till genetiska algoritmer, kan förbättra prestandan hos den genetiska algoritmen i syfte att lösa ett problem. I alla testade problem gav den nya genetiska algoritmen bättre resultat i början och förbättrade även resultatet med tiden.

För att testa upptäckten och förhoppningsvis ge mer bevis som stöd åt upptäckten, konstruerade forskarna femtio olika TSP-problem som sekventiellt testades genom det nya systemet (GA-modul och CBR-modul). Desto fler problem som behandlades, desto bättre lösningar gav det nya systemet. Lösningarna var blev bättre och bättre redan vid första generationen och det tog kortare tid att skapa de högkvalitativa lösningar, som tidigare tagit mycket längre tid.

Forskarna skriver avslutningsvis att det finns mycket kvar att undersöka inom de genetiska algoritmerna och att arbeta med en så kallat CBR-modul, i syfte att förbättra en genetisk algoritm. För att förbättra denna framarbetade lösningen, går det att forska och experimentellt testa olika typer av crossover, mutationer och selektion. Även CBR-modulen och dess funktioner går att förändra, för att anpassas till att lösa TSP-problem.

3 Metod

För att göra forskningen möjlig krävs metoder och tekniker för att hantera planeringen av arbetet, insamling av information och för att analysera det insamlade materialet. I följande kapitel skall uppsatsens arbetssätt klargöras och motiveras.

3.1 Forskningsansats

I denna uppsats avser jag att undersöka hur genetiska algoritmer fungerar i teorin. Jag avser även att undersöka hur genetiska algoritmer tillämpas mot the traveling salesman problem, som är ett optimeringsproblem som beskrivits tidigare i kapitel 2.6. Min studie kommer på ett explorativt sätt att leta efter egenskaper hos genetiska algoritmer, som kan vara av betydelse för en utvecklare, som är i behov av att välja en lämplig lösningsteknik för någon form av problemlösning. Jag avser även att utveckla en modell över en genetisk algoritm, som behandlar ett specifikt optimeringsproblem. Modellen skall sedan användas manuellt, för att demonstrera hur modellen är tänkt att fungera. Jag kommer att välja lämpliga metoder för att på bästa sätt kunna besvara uppsatsens forskningsfråga utifrån den ovanstående ansatsen.

3.2 Val av metoder

Valet av metoder och arbetssätt för denna uppsats bygger till stora delar på forskningsmetodik beskriven av Blaxter L., Hughes C. & Tight M., (2001). För att genomföra forskning och välja vilken metod som passar bäst delas forskningsmetodiken först in i tre delar: forskningsfamilj, forskningens angreppssätt och forskningsteknik. (Blaxter L., Hughes C. & Tight M., 2001) Kapitlet kommer även att innehålla viss kompletterande forskningsteori utöver Blaxter L., Hughes C. & Tight M., (2001).

3.3 Forskningsfamilj

Beroende på det syfte denna uppsats har och den forskningsfråga denna uppsats vill undersöka, anser jag att det är lämpligt att välja en kvalitativ forskningsfamilj. Ett kvalitativt angreppssätt koncentrerar sig på att samla in och analysera data på många olika sätt. Detta angreppssätt riktar in sig på att upptäcka och på ett explorativt sätt analysera data i djupare detalj. Angreppssättet försöker skapa ett djup i forskningen till skillnad från ett kvantitativt angreppssätt, som ofta skapar en bredd. (Blaxter L., Hughes C. & Tight M., 2001) För att ha möjlighet att kunna dra slutsatser om hur genetiska algoritmer fungerar och tillämpas är det av intresse att undersöka de genetiska algoritmernas egenskaper på ett kvalitativt sätt och inte ett kvantitativt sätt.

Blaxter L., Hughes C. & Tight M., (2001) anser även att vid valet av forskningsfamilj, skall det även tas beslut om arbetet kommer att vara inriktad på fältarbete, exempelvis empiriska studier i form av intervjuer eller observationer ute på fält, eller om arbetet främst kommer att handla om skrivbordsarbete. Mitt val i denna fråga faller på skrivbordsarbete. Detta val bygger på att jag vill undersöka genetiska algoritmer i teorin och även hur genetiska algoritmer tillämpas mot ett specifikt problemområde. Valda tillämpningar avser jag att studera genom en dokumentstudie, vilket bygger på att jag i förarbetet till denna uppsats har hittat ett antal artiklar om genetiska algoritmer som tillämpas mot the traveling salesman problem. I tron om att dessa artiklar innehåller tillräckligt med material för att kunna förstå hur genetiska algoritmer används och tillämpas, anser jag att jag inte har mycket att vinna på att utföra någon form av fältarbete.

3.4 Forskningens angreppssätt

Val av angreppssätt ligger mellan ett antal olika alternativ enligt Blaxter L., Hughes C. & Tight M., (2001). Samma författare beskriver fallstudier enligt följande: ”fallstudie är det angreppssätt som är lämpligt att välja, om det fenomen som vill studeras inte skiljer sig eller utmärker sig i sin kontext”. Detta tycker jag överensstämmer relativt bra med den undersökning jag vill genomföra. Min forskning kommer att inrikta sig på att undersöka genetiska algoritmer i teorin och befintliga tillämpningar inom ett specifikt problemområde, nämligen the traveling salesman problem. Relevansen av undersökningen ligger i att identifiera de egenskaper som är specifika för de genetiska algoritmerna och dessa så kallade ”fenomen” ligger dolda i de tillämpningar som skall undersökas samtidigt som de inte utmärker sig från sin kontext. En fallstudie skulle alltså möjliggör att identifiera och definiera de egenskaper som finns dolda i det kontext, som försöker beskriva hur ett problem är löst med någon lösningsteknik. Istället för att genomföra en fallstudie har jag valt att studera tillämpningar av genetiska algoritmer, genom att studera artiklar som beskriver dessa tillämpningar.

Blaxter L., Hughes C. & Tight M., (2001) beskriver ett antal andra angreppssätt. Ett av de många angreppssätten är experiment. Att utföra experiment där de genetiska algoritmernas egenskaper och möjligheter att tillämpas mot olika problemområden skulle vara mycket intressant att konstruera och utföra. Jag upplever dock att ett sådant experiment skulle bli allt för tidskrävande. Detta i kombination med att jag upplever mig själv endast inneha bristfälliga kunskaper inom några av de kunskapsområden, som krävs för att kunna konstruera och genomföra ett experiment av den typen. Detta medför att valet att utföra experiment, som angreppssätt i denna uppsats, måste förkastas.

Jag skall även försöka visa med hjälp av en modell, hur genetiska algoritmer skulle kunna användas i ett beslutsstödjande system. Modellen skall även delvis testas för att demonstrera hur den genetiska algoritmen arbetar inom det specifika problemet.

Styrkor med att arbeta med dokumentstudier av den här typen, är att de baseras på människors arbete och erfarenhet inom ett visst område och speglar ofta verkligheten på

ett bra sätt. Denna typ av studie lämpar sig även bra då specifik egenskap inom ett fall skall generaliseras och placeras på en högre allmän nivå. (Blaxter L., Hughes C. & Tight M., 2001). Dessa fördelar är något som känns viktigt i denna forskning. Då jag vill ge en bild av verkligheten, analysera de specifika iakttagelserna och dra slutsatser om vilka andra typer av optimeringsproblem genetiska algoritmer går att tillämpa på, i syfte att skapa fungerande beslutsstöd.

Det Blaxter L., Hughes C. & Tight M. (2001) nämner som nackdel med denna typ av studie är att komplexiteten i specifika fall kan göra att analysarbetet blir väldigt svårt. De menar dock att forskare bör vara och ofta är medvetna om kopplingar mellan olika händelser, variabler och skeenden inom det specifika området. Forskaren kan på så sätt undvika den komplexitet som kan uppkomma i och med analysarbetet.

3.5 Forskningsteknik

Vid val av forskningsteknik och insamlingsmetod faller valet på en dokumentstudie i samband med ett skrivbordsarbete istället för fältarbete. Detta val har redan motiverats ovan (se kapitel 3.3). Alla forskningsarbeten inkluderar någon form av dokumentstudie, där forskaren läser, förstår och kritiskt analyserar, det andra redan har skrivit (Blaxter L., Hughes C. & Tight M., 2001). Genom att genomföra en dokumentstudie gentemot dokument som beskriver tillämpningar som behandlar the traveling salesman problem, avser jag att samla in tillräckligt med material, för att kunna uppfylla mitt syfte och besvara uppsatsens forskningsfråga.

Då jag kommer att arbete med statiskt material, kommer jag inte att begränsas av mina ställda frågor och mina kunskaper vid exempelvis en intervju, samtidigt som jag inte kommer att ha möjlighet till att ställa följdfrågor eller förklarande frågor. Detta innebär både fördelar och nackdelar, vilket kommer att diskuteras vid en senare metoddiskussion. Genom en dokumentstudie kommer jag direkt att kunna studera vad experter på området har skrivit och sagt om det problemområde jag vill undersöka.

3.6 Litteraturstudie

Det är möjligt att genomföra forskning utan att läsa, men det är en väldigt ovanlig väg att gå. Genom att läsa litteratur, forskningsrapporter och aktivt söka bland skriftlig information, skapar forskaren sig bakomliggande kunskap inom olika ämnesområden och tilllåter sig själv att vidga sin syn för olika företeelser. (Blaxter L., Hughes C. & Tight M., 2001)

Blaxter L., Hughes C. & Tight M. (2001) menar även att det starkt rekommenderas att forskning på en högre akademisk nivå skall bygga på teorier, undersökningar och fakta, hämtat från skrivna verk. Detta för att skapa en trovärdighet i forskningen, genom att referera till andra forskares verk och på detta sätt stärka sin egen forskning. Det finns många olika källor för att söka information från, bland dessa finns publicerat material av

olika slag, men även opublicerat material går att använda sig av. Det kan röra sig om rapporter, som endast har cirkulerat inom ett företag och inte publicerats för allmänheten. Ofta finns det ett intresse att läsa litteratur som är färsk och innehåller uppdaterad information om ett ämnesområde. Samtidigt som det kan vara intressant att referera till klassisk forskning inom ämnesområdet, eftersom dessa verk ofta har en högre trovärdighet. Det är även av intresse för en forskare att inte bara läsa teoretiskt material inom ämnesområdet, utan även att läsa om olika metoder och arbetssätt. (Blaxter L., Hughes C. & Tight M., 2001)

Till denna uppsats har jag helt och hållet valt att genomföra min forskning genom att studera litteratur och skriva beskrivningar av tillämpningar av genetiska algoritmer, i syfte att samla in information. Litteratur om problemlösning, beslutsfattande och beslutsstödjande system har lånats från LTU:s bibliotek. Även teoriböcker om forskningsmetodik har lånats från biblioteket. För att hitta mer problemspecifik litteratur, som behandlar genetiska algoritmer och the traveling salesman problem, har Internet använts och jag har sökt information på diverse olika artikel databaser, exempelvis Ebsco. Ett antal artiklar, som beskriver tillämpningar av genetiska algoritmer, har studerats och de två artiklar jag funnit mest intressanta har jag därefter valt att presentera i denna uppsats teorikapitel. De både tillämpningarna angriper the traveling salesman problem med hjälp av genetiska algoritmer. Artiklarna visar samtidigt hur tillämpningar av genetiska algoritmer kan skiljas sig från varandra. De två artiklarna har valts med omsorg och med hänsyn till att de skall vara på en hög akademisk nivå.

3.7 Analysmetod

Insamlad data kan finnas i olika former och i mitt fall kommer insamlad data ursprungligen från teorin och utvalda artiklar, som behandlar genetiska algoritmer och deras tillämpningar på the traveling salesman problem. De utvalda artiklar presenterar i kapitel 2.7, där jag presenterar min tolkning av artiklarna, deras bakgrund, arbetssätt, metod och resultat. Utifrån en kvalitativ analysmetod kan forskare analysera data av den här typen, för att få djupare förståelse för innehållet i insamlad data och bakomliggande faktorer (Blaxter L., Hughes C. & Tight M., 2001).

Enligt Blaxter L., Hughes C. & Tight M., (2001) är planering ett förarbete till analysfasen som är mycket viktig för att möjliggöra en tillfredställande analys. Vid analys skall jag söka svar på vissa förutbestämda frågeställningar, vilket delvis kan liknas med en typ av guide som annars används vid intervjuer. Då jag har läst teori och beskrivningar av tillämpningar där genetiska algoritmer används, har jag funnit följande frågeställningar intressanta att undersöka, för att kunna förmedla en bild över hur genetiska algoritmer arbetar mot ett valt optimeringsproblem och även kunna svara på denna uppsats forskningsfråga.

- Möjligheter och begränsningar: Denna frågeställning avser att söka uttalade möjligheter med användning av genetiska algoritmer. Jag avser även att söka begränsningar, svårigheter och brister vid användning av genetiska algoritmer.

- Uppbyggnad av individer: Jag avser att undersöka hur de genetiska algoritmernas individer är uppbyggda, eftersom jag i mina teoretiska studier har sett att presentation/uppbyggnaden av individer ser olika ut och är beroende av vilket specifikt problem individen skall vara verksam mot.

- Anpassningar av selektion: Teorin nämner ett antal olika metoder för att genomföra selektion. Val av selektionsoperator tycks variera mellan olika problem och jag avser att undersöka om så är fallet och vad det resulterar i att anpassa selektionsoperatorn till problemet.

- Anpassning av crossoveroperatorer: Ovan nämns anpassning av selektion och denna frågeställning ställs av samma anledning.

- Anpassning av mutationsoperatorer: Det finns även olika typer av mutationsoperatorer enligt teorin och jag vill undersöka om och varför även mutationsoperatorn anpassas till det specifika problemet.

- Prestanda: Jag vill skapa mig en bild över de genetiska algoritmernas prestanda och hur de presterar mot the traveling salesman problem.

- Skillnader mot teorin: Jag avser även att söka efter avvikelser från vad teorin säger, mot hur genetiska algoritmer fungerar i de två tillämpningarna mot the traveling salesman problem.

- Andra upptäckter: Ovanstående frågeställningar är framtagna i avseende att fungera som sorts guide över vad jag skall söka efter i analysarbetet. Jag vill dock inte helt låsa mig vid dessa frågeställningar, utan även andra upptäcker som är intressanta skall så klart även dem lyftas fram.

Analysen skall utföras mot min tidigare presenterade teori, men främst de två studerade tillämpningarna av genetiska algoritmer mot the traveling salesman problem. Bland detta material skall de olika frågeställningarna undersökas och besvaras om möjligt. Jag är medveten om att det är möjligt att vissa frågor inte kommer att gå att besvara, men i så stor grad som möjligt skall jag följa ovanstående frågeställningar. Analysen skall sedan sammanfattas och leda till att jag identifierar ett nytt optimeringsproblem, förekommande i ett beslutsstödjande system. Jag avser därefter att utveckla en modell, som visar ett exempel på hur genetiska algoritmer kan tillämpas inom denna typ av beslutsstöd. Därefter skall upptäckter i och med mitt arbete diskuteras, för att jag skall ha möjlighet att dra slutsatser och ge svar på uppsatsens forskningsfråga.

3.8 Validitet och reliabilitet

Jag har valt att utföra dokumentstudier som insamlingsmetod med en kvalitativ ansats. Två valda artiklar skall analyseras utifrån ett antal frågeställningar, som sedan besvaras och diskuteras. Enligt Blaxter L., Hughes C. & Tight M., (2001) ger denna kvalitativa ansats forskaren möjlighet att upptäcka och på ett explorativt sätt analysera data i djupare detalj. Detta medför att jag kan få djupare förståelse för hur genetiska algoritmer fungerar mot ett specifikt problem. Denna ökade förståelse ger därför denna undersökning mer trovärdighet och en högre validitet och reliabilitet.

Eftersom dokumentstudien begränsas mot the traveling salesman problem och två artiklar, som tillämpar genetiska algoritmer mot problemet, kan litteraturstudien endast anses ge god validitet och reliabilitet inom dessa gränser. Detta bör läsaren till uppsatsen ha i åtanke, då mitt redovisade resultat endast är baserat mot vad teorin säger, hur beskriva tillämpningar mot TSP har sett ut och mina erfarenheter av att utveckla en genetisk algoritm.

Min analys kommer att mynna ut i ett problem, som behandlas i ett beslutsstödjande system och angrips med hjälp av genetiska algoritmer. Jag avser att utveckla och testa en modell över en genetisk algoritm, som visar hur en genetisk algoritm skulle kunna användas för att angripa en annan typ av optimeringsproblemet. Genom detta arbetssätt avser jag att skapa en viss bredd då jag kombinerar teoretiska studier med att själv ha arbetat med att ta fram en genetisk algoritm. På detta sätt vill jag öka trovärdigheten i mina upptäckter.

4 Analys

I följande kapitel skall tidigare presenterad teori och de presenterade tillämpningarna analyseras, enligt tidigare presenterad analysmetod. Analysen skall leda till insikt i hur genetiska algoritmer och identifiera viktiga faktorer i samband med att arbete med genetiska operatorer. Analysen skall sedan mynna ut i ett förslag på en annan typ av optimeringsproblem, vilket jag senare kommer att angripa genom att utveckla en modell för att behandla det valda problemet.

4.1 Frågor

- **Möjligheter och begränsningar**: Artikeln skriven av Tsai H., Yang J., Tsai Y. & Kao C. (2004), visar på att genetiska algoritmer, med anpassade genetiska operatorer, kan hitta den optimala lösningen för TSP-problem med under sextusen noder och genomsnittliga felgränsen är 0,0074 procent från den optimala lösningen på det största undersökta problemet (13509 noder), som ingick i experimentet. Författarna anser utifrån detta att genetiska algoritmer kan ses som en robust lösningsteknik för att lösa TSP-problem inom en acceptabel tidsåtgång.

  Li G. & Louis S., (2000) använder sig av ett lite annorlunda angreppssätt för att förbättra och anpassa en genetisk algoritm. Istället för att anpassa de genetiska operatorerna, väljer författarna att arbeta med CBR, för att på detta sätt underlätta för den genetiska algoritmen. CBR injicerar initial populationer som bygger på kunskap lagrat i CBR-modulen. Det visar sig att detta angreppssätt skapar en genetisk algoritm, som sparar tid och även genererar bättre lösningar.

  Detta angreppssätt borde rimligtvis även gå att föra över till andra problem och inte bara fungera mot TSP-problem. En annan tanke är att en CBR-modul ytterligare skulle kunna förbättra HeSEA som diskuterades i artikeln av Tsai H., Yang J., Tsai Y. & Kao C., (2004). Samtidigt som det är av vikt att inse att utvecklingen av en CBR-modul rimligtvis innebär merarbete, vid utvecklingen av en lösningsmetod.

  För att återgå till artikeln skriven av Tsai H., Yang J., Tsay Y. & Kao C. (2004) tycks en begräsning vid användning av genetiska algoritmer vara att den använda genetiska algoritmen inte alltid hittade den optimala lösningen på de olika problemen. Det kan vara svårt att veta om den genetiska algoritmen ger en optimal lösning eller om lösningen går att göra ännu bättre då den optimala rutten är okänd.

  Det finns många olika operatorer att använda i en genetisk algoritm (Tsai H., Yang J., Tsai Y. & Kao C., 2004), detta kan skapa svårigheter för en användare att välja och använda en lämplig operator för ett specifikt problem. En möjlighet

kan vara att testa olika genetiska operatorer mot det aktuella problemet och på detta sätt hitta den mest lämpliga operatorn. Detta innebär dock extra arbete.

Genetiska algoritmer kan även vara väldigt tidskrävande. De tre största TSP-problemen, som angripits med HeSEA tog väldigt lång tid, utan att hitta den optimala rutten vid varje tillfälle. Till det största problemet hittades inte den optimala rutten. Det tredje största problemet med 4461 noder tog i genomsnitt ~39 minuter och då hittades den optimala rutten sexton av tjugo gånger. Dock kan resultatet mot problemet ”vm1084” med 1084 noder anses som bra och även snabbt utfört. Till problemet hittades den optimala rutten tjugo av tjugo gånger, på en genomsnittlig tid på en minut och tjugo sekunder. Detta tyder på att komplexiteten på problemet kan påverka arbetstiden till väldigt stor del.

- **Uppbyggnad av individer**: Li G. & Louis S., (2000) säger att det behövs genomföras förändringar mot traditionella genetiska algoritmer för att hantera TSP-problem. Det går inte att använda binära kromosomer för att representera individer, utan på grund av att individen skall representera en sekventiell lösning är en individ uppbyggd på följande sätt:

  [startnod(0), besökt nod(1), besökt nod(2), besökt nod(3), …. besökt nod(n)]

  En individ är alltså uppbyggt som en vektor bestående av ett antal element, som motsvara de besökta städerna i tur och ordning. Om det finns fyra städer där kan exempelvis en rutt vara representerad på följande sätt: [1, 3, 4, 2]. Vilket innebär att rutten startade i stad 1, sedan vidare till stad 3, 4 och 2 för att sedan återvända till stad 1 igen.

  Tsai H., Yang J., Tsai Y. & Kao C., (2004) beskriver aldrig uppbyggnaden av individer eller kromosomer, utan anger endast att en population består av ett antal individer vilket kortfattat även är det teorin och Li G. & Louis S., (2000) säger.

  Utifrån detta går det att se att uppbyggnaden av individer kan variera mellan olika problem och att utvecklaren av en genetisk algoritm, måste beakta vilken uppbyggnad av individerna som lämpar sig för det aktuella problemet.

- **Anpassningar av selektion**: Tsai H., Yang J., Tsai Y. & Kao C., 2004 beskriver selektionsprocessen i HeSEA på följande sätt: Varje individ testas genom en fitnessfunktion och genom heterogen parselektion väljs familjefadern och den individ som har mest gemensamt med familjfadern utifrån kantlikheter i det specifika TSP-problemet. Dessa två individer blir föräldrar till EAX-operatorn som skall generera ett antal barn. Men det slutgiltiga urvalet av vilken individ, som går vidare till nästa generation är ännu inte slutfört. Det barnet med den kortaste färdsträckan väljs sedan ut och LK-operatorn utför mutation på det genererade barnet. Detta upprepas och den förälder eller barn, som har det bästa fitnessvärdet överlever och får gå vidare till nästa generation.

Selektionen är alltså anpassad specifikt för TSP och bygger delvis på heterogen parselektion, EAX och LK. Att dessa operatorer var lämpade för TSP kom forskarna fram till genom experiment över potentiella crossoveroperatorer och mutationsoperatorer. Selektionen bygger även på ”bra kanter” i TSP-problemet, vilket författarna menar är mycket viktigt för att kunna lösa TSP-problem. Operatorer för att lösa TSP-problem skall vara kantbaserade och innehålla goda mekanismer för att skapa och bibehålla bra kanter (Tsai H., Yang J., Tsai Y. & Kao C., 2004).

Li G. & Louis S., (2000) använder sig istället av den traditionell roulettehjulsselektion, på samma sätt som teorin beskriver den. Vid denna typ av selektion är det dock inte säkert att den individen med högst fitness överlever. De individerna med högst fitness har dock större chans att ingå i urvalet är individer med lägre fitness.

- **Anpassning av crossoveroperatorer**: I forskningen av Tsai H., Yang J., Tsai Y. & Kao C. (2004) testades fyra olika crossoveroperatorer, där en operator kallad EAX visade sig ge bäst resultat mot TSP. Utifrån detta resultat väljer forskarna att använda sig av EAX vid utveckling av HeSEA. Forskarna menar att EAX är en crossoveroperator som kan skapa och bibehålla så kallade bra kanter i ett TSP-problem. Forskarna menar att detta är ett måste för att skapa en genetiska algoritm, som på ett effektivt sätt kan lösa stora TSP-problem.

  Detta tyder på att det finns ett antal varianter på hur en crossoveroperator kan se ut, vilket skiljer sig en del från vad studerad teori beskriver. Det går att antaga att

4.2 Sammanfattning av analysen

Analysen ovan visar tydligt att de genetiska operatorerna inte alltid ser ut på samma sätt i teorin som i verkligheten. Innan jag började skriva denna uppsats trodde jag att kunskap om teoretiska genetiska algoritmer, skulle ge mig ett kraftigt verktyg för att angripa ett stort antal olika problemområden. Efter att jag nu har studerat tillämpningar av genetiska algoritmer mot the traveling salesman problem, har jag upptäckt att det är mer komplicerad än jag trodde att använda sig av genetiska algoritmer och att teoretiska genetiska algoritmer inte kan angripa vilket problem som helst.

De teoretiska genetiska algoritmerna innehåller endast förslag på genetiska operatorer, i syfte att förklara hur dessa operatorer är tänkta att fungera och vad deras syften är. Till specifika problem måste dessa genetiska operatorer anpassas av olika anledningar. Dels kan det annars leda till att otillåtna individer skapas (Li G. & Louis S., 2000), vilket inte nödvändigtvis förstör hela försöket med att angripa problemet med genetiska algoritmer. De otillåtna individerna skulle kunna förkastas vid selektion och problemet är på detta sätt löst. Det finns dock en chans att alla individer i en population blir otillåtna individer och då skulle det innebära att evolutionen och utvecklingen i den genetiska algoritmen skulle stanna upp. Genom att anpassa de genetiska operatorerna kan användaren undvika sådana fallgropar.

Anpassning av genetiska operatorer tycks även vara nödvändigt för att skapa en effektiv och snabb genetisk algoritm. De både analyserade tillämpningarna försöker skapa denna effektivitet genom två olika angreppssätt. Dels genom att anpassa crossoveroperatorn, mutationsoperatorn och selektionen och kombinera dessa till en nyutvecklad algoritm, som fick namnet HeSEA. Li G. & Louis S. (2000) försökte däremot skapa en effektiva genetisk algoritm genom att tillföra en CBR-modul. Denna modul skulle ta vara på information om tidigare fall och lösningar, i syfte att kunna injicera anpassade initial population vid nya TSP-problem.

Tsai H., Yang J., Tsai Y. & Kao C., (2004) säger att flera olika problem går att uttrycka som ett TSP-problem och på detta sätt är det rimligt att dra slutsatser om att dessa problem går att lösa med hjälp av genetiska algoritmer, vilket tyder på att genetiska algoritmer är lämpligt att tillämpa mot flertalet optimeringsproblem. Utifrån att ha studerat genetiska algoritmer och tillämpningar av genetiska algoritmer mot TSP-problem kan jag se vissa typer av optimeringsproblem, som kan lösas med genetiska algoritmer. Jag ser dock även flera problem som kan uppstå i och med användning av denna lösningsteknik.

Individer kan se olika ut, men i en teoretiska genetiska algoritm är individer uppbyggda som vektorer, där olika värden symboliserar något i det specifika problemet. Till TSP-problemet låter Li G. & Louis S., (2000) en vektor representera den färdväg som skall motsvara lösningen till problemet. Varje plats i vektorn motsvar en stad, vilket i tur och ordning blir besökt. Det är här risken för otillåtna individer uppkommer och det måste motverkas med hjälp av anpassning av de genetiska operatorerna. I andra exempel

representeras individerna av en vektor med olika värden, där de olika värdena i sin tur står för en funktion i något programmeringsspråk (Connoly B., 2004).

Arbetsgången i denna uppsats var att undersöka hur genetiska algoritmer fungerar och även analysera två tillämpningar av genetiska algoritmer mot ett verkligt problem. Detta har jag gjort och nästa steg är att identifiera ett nytt optimeringsproblem, där jag tycker mig se tendenser till att genetiska algoritmer skulle gå att använda för att behandla det specifika problemet. Jag har ägnat tid åt att läsa böcker av Marakas G. (2003) och Aronson J. & Turban E. (2001) i syfte att identifiera kända och uttalade optimeringsproblem, som jag tror kan lösas med hjälp av genetiska algoritmer. Jag har även funderat över olika problemområden utifrån egna erfarenheter och jag har upplevt att många problem är svåra att behandla med genetiska algoritmer av olika anledningar. Problem uppstår exempelvis vid uppbyggnaden av individer och de olika genetiska operatorerna är svåra att skapa, då problemområden kan se väldigt olika ut och innehåller många olika egenskaper, som kan vara mycket svåra att skapa i en genetisk algoritm. Jag vill dock inte påstå att det är en omöjlighet att använda genetiska algoritmer i något specifikt problem. Detta kan bero på att jag saknar tillräckligt djupa kunskaper om programmering och hur genetiska algoritmer ser ut i programmeringskod.

Ett problem jag identifierat, som jag anser ska gå att lösa med hjälp av genetiska algoritmer, är hämtat ifrån mina tidigare studier på C-nivå. I kursen ”IED331 - Intelligenta beslutsstödjande system” genomfördes ett projektarbete där det förekom ett behov av ett beslutsstöd för att bestämma den produktionsmix, som gav högsta vinst för företaget. Jag har analyserat olika egenskaper i ett sådant problem och vilka behov det skulle kunna finnas i ett beslutsstöd av den typen.

Valet av det optimeringsproblem jag avser att angripa med genetiska algoritmer, är alltså ett produktionsproblem. Detta problem och deras egenskaper beskrivs i nästa kapitel.

5 Ett produktionsproblem angrips med genetiska algoritmer

I detta kapitel skall ett produktionsproblem angripas med genetiska algoritmer. Problemet bygger på ett produktionsproblem som förekommit i en tidigare genomförd kurs vid LTU. En problembeskrivning kommer att förklara problemets bakgrund och egenskaper. Därefter skall jag förklara vilka svårigheter som föreligger i och med att angripa problemet med genetiska algoritmer. Efter detta skall jag utveckla och anpassa de genetiska operatorerna efter behov och slutligen skall en modell över den genetiska algoritmen presenteras.

5.1 Beskrivning av produktionsproblemet

Idén till det valda produktionsproblemet bygger på en kurs jag tidigare läst vid LTU. Problemet är inte en exakt kopia av det problemet, men jag har tänkt mig problemet i en viss situation där det är verksamt i samband med ett beslutsstöd, som även innehåller andra funktioner. Problemet handlar om ett företag som tillverkar produkter. De har en bred produktkatalog med cirka tusen olika produkter. Vissa har funnits med under de senaste tio åren, medan andra produkter är nyare. Problemsituationen för företaget är att de skall ta beslut över hur stor kvantitet av de olika produkterna som skall produceras för att nå högst avkastning.

De olika produkterna har ett försäljningspris och en produktionskostnad. Eftersom företaget sedan tidigare har ett beslutsstödjande system, finns det en funktion där all historik om tidigare års försäljning har lagrats. Denna information kan analyseras och bearbetas för att ge uppgifter om hur stor kvantitet av respektive produkt det är sannolikt att företaget har möjlighet att sälja under den kommande perioden - årsvis, per kvartal eller månadsvis.

Företaget litar starkt på uppgifterna om den sannolika kommande försäljningen, som beslutstödet presenterar. Företaget avser att aldrig producera mer än den angivna sannolika försäljningen av en produkt, detta eftersom de inte vill lagerhålla produkter under någon längre tidsperiod.

Företaget har även en budget över hur mycket pengar som skall satsas på tillverkningen av produkterna och alla kostnader är inräknade i produktionskostnaden för varje produkt. Företaget vill alltså få förslag över vilka produkter som skall produceras och hur många av varje typ av produkt, för att ge en så hög avkastning som möjligt. Problemet med att göra denna planering manuellt ligger i att den beräknade försäljningen varierar kraftigt mellan olika produkter och att även försäljningspris och produktionskostnader varierar kraftigt. Enklare blir det inte då produktionskatalogen innehåller cirka tusen produkter.

5.2 Utveckling av den genetiska algoritmen

En genetisk algoritm innehåller representation av individer, en fitnessfunktion, selektion, crossover och mutation. Alla dessa funktioner skall ingå i den genetiska algoritm jag skall utveckla för att angripa det angivna produktionsproblemet.

Representation av individer kan enligt Connoly B., (2004) och Li G. & Louis S., (2000) se väldigt olika ut. I detta fall väljer jag att skapa en vektor med en cell för respektive produkt som finns i produktkatalogen. Värdet i respektive cell, visar på antalet produkter av den typen som skall produceras för den specifika individen, vilket motsvarar en föreslagen lösning. Eftersom det finns uppgifter om den förväntade försäljningen för respektive produkt och företaget inte vill producera fler produkter än vad dessa uppgifter pekar på började jag fundera på hur representationen skulle gå till. Första tanken var att låta varje cell innehålla ett heltal som enkelt stod för antalet produkter som skall produceras av den typen. Eftersom detta skulle kunna leda till att en viss produkt producerades i ett större antal än den förväntade försäljningen skulle det bryta mot företagets önskemål. Det skulle alltså innebär att en otillåten individ skulle kunna genereras, vilket inte är önskvärt.

För att lösa detta problem kom idén om att låta ett värde mellan 0-100 representera antalet produkter som ska produceras. Värdet motsvarar en procentuell del av den förväntade försäljningen för produkten, där 100 motsvarar att de skall produceras lika många produkter av den typen, som den förväntade försäljningen pekar på. Värdet 50 motsvarar att 50 procent av antalet produkter, som förväntas sälja av den typen, skall produceras. På detta sätt genereras inga otillåtna individer enligt det problem som beskrevs ovan.

Det finns dock andra problem, vilket kan medföra att otillåtna individer genereras. Företaget har en budget över hur mycket de är redo att investera i totala produktionskostnader. En individ kan lätt genereras på ett sådant sätt att produktionskostnaden överstiger den önskade budgeten och en otillåten individ har uppkommit. Detta måste lösas på något sätt och enligt denna genetiska algoritm blir det en mutationsoperator som tar hand om detta problem. Operatorn kommer att beskrivas senare i kapitlet.

Hittills har jag beskrivit hur representationen av individen skall se ut för att undvika otillåtna individer. För detta behövs en funktion för att omkoda individerna från procentuella värden till heltal. Det behövs även en fitnessfunktion till denna genetiska algoritm och den byggs upp på ett sådant sätt att den beräknar den totala produktionskostnaden för den totala lösningen och den totala vinsten, efter att ha antagit att uppgifterna om den förväntade försäljningen stämmer. Den siffra som beräknas är alltså den totala avkastningen. Detta fitnessvärde står i direkt relation till det mål företaget vill åstadkomma med hjälp av detta beslutsstödjande system.

Selektion nämner Tsai H., Yang J., Tsai Y. & Kao C., (2004) som något mycket viktigt för att kunna lösa ett TSP-problem. Jag ser dock inte samma behov i detta

produktionsproblem, även om det är möjligt att en smartare selektion kan skapa bättre prestanda för den genetiska algoritmen. Jag väljer dock en vanlig typ av selektion, som bygger på roulettehjulsprincipen. De individer med högst fitness har större chans att ingå i urvalet, men när roulettehjulet snurrar finns det dock en chans att de individer med högst fitness inte blir valda. I ett längre tidsperspektiv är dock urvalet i direkt proportion mot individernas fitnessvärde (Olsson B., 1996).

När ett urval har skett är det enligt teorin oftast en crossoveroperator som sätts i bruk i syfte att förändra populationen på något sätt. Tsai H., Yang J., Tsai Y. & Kao C., (2004) använder EAX som crossoveroperator och nämner ett antal andra i sin forskning. (Olsson B., 1996) beskriver tvåpunkts-crossover och även om detta är en enkel crossoveroperator tycker jag att den fyller sin funktion i detta produktionsproblem. När det handlar om prestanda, gäller samma antagande i detta fall. En annan eller en anpassad crossoveroperator skulle möjligen kunna öka prestandan i den genetiska algoritmen. Då jag inte har någon möjlighet att testa olika operatorer är det dock omöjligt att säga något med säkerhet.

Tvåpunkts-crossover går ut på att det slumpas två punkter i respektive förälder och värdena inom dessa två punkter byter helt enkelt plats (Olsson B., 1996). Detta skapar en variation mellan individerna, samtidigt som det kan generera otillåtna barn. För att ta hand om detta hänvisar jag än en gång till mutationsoperatorn, som i den här genetiska algoritmen får en mycket viktig roll.

Det som återstår är mutationsoperatorn som i den här genetiska algoritmen kommer att bli något mer omfattande än andra teoretiska mutationsoperatorer. En funktion som är nödvändig är att kunna förändra en individ om det visar sig att individens totala produktionskostnad överstiger den givna budgeten. I mutationsoperatorn anser jag det vara lämpligt att ha en funktion som kan beräkna om individen motsvarar ett överskott eller ett underskott gentemot budgeten. Dessa två funktioner beräknas på följande sätt:

- Om den totala produktionskostnaden är mindre än budgeten → Överskott
- Om den totala produktionskostnaden är större än budgeten → Underskott

Mitt förslag är att utifrån dessa två förutsättningar, anpassas individen på två olika sätt. De två nya funktionerna döper jag av naturliga skäl till ”överskott” och ”underskott”.

Funktionen ”Överskott” är till för att utöka individen då det finns pengar över till att producera flera produkter av någon typ. Om denna funktion aktiveras slumpas en position i den aktuella individen. Därefter beräknas antalet produkter av den typ som positionen representerar, som budgeten tillåter att det ytterligare produceras. För att undvika att det produceras ett större antal av en produkt, än vad den beräknade försäljningen tyder på, måste något göras åt detta. Enkelt kan detta undvikas genom att en position i en individ aldrig kan överstiga 100 (alltså 100 procent av den förväntade försäljningen).

Funktionen ”Underskott” har i uppgift att förändra en individ då den totala produktionskostnaden överskrider den totala budgeten. En position i den aktuella

individen slumpas och därefter sker följande beräkning. Det beräknade underskottet delas med kostnaden för att tillverka en enhet av den aktuella produkten. Resultatet avrundas uppåt till närmaste heltal. Därefter tar mutationsoperatorn bort det beräknade antalet från den aktuella positionen. Om individen, på den aktuella positionen, inte består av ett värde större eller lika stort som det beräknade antalet som skall tas bort. Då tar mutationsoperatorn bort det antalet som finns och funktionen ”Underskott” upprepas. I och med att funktionen ”Underskott” upprepas kommer individen slutligen att bli en tillåten individ.

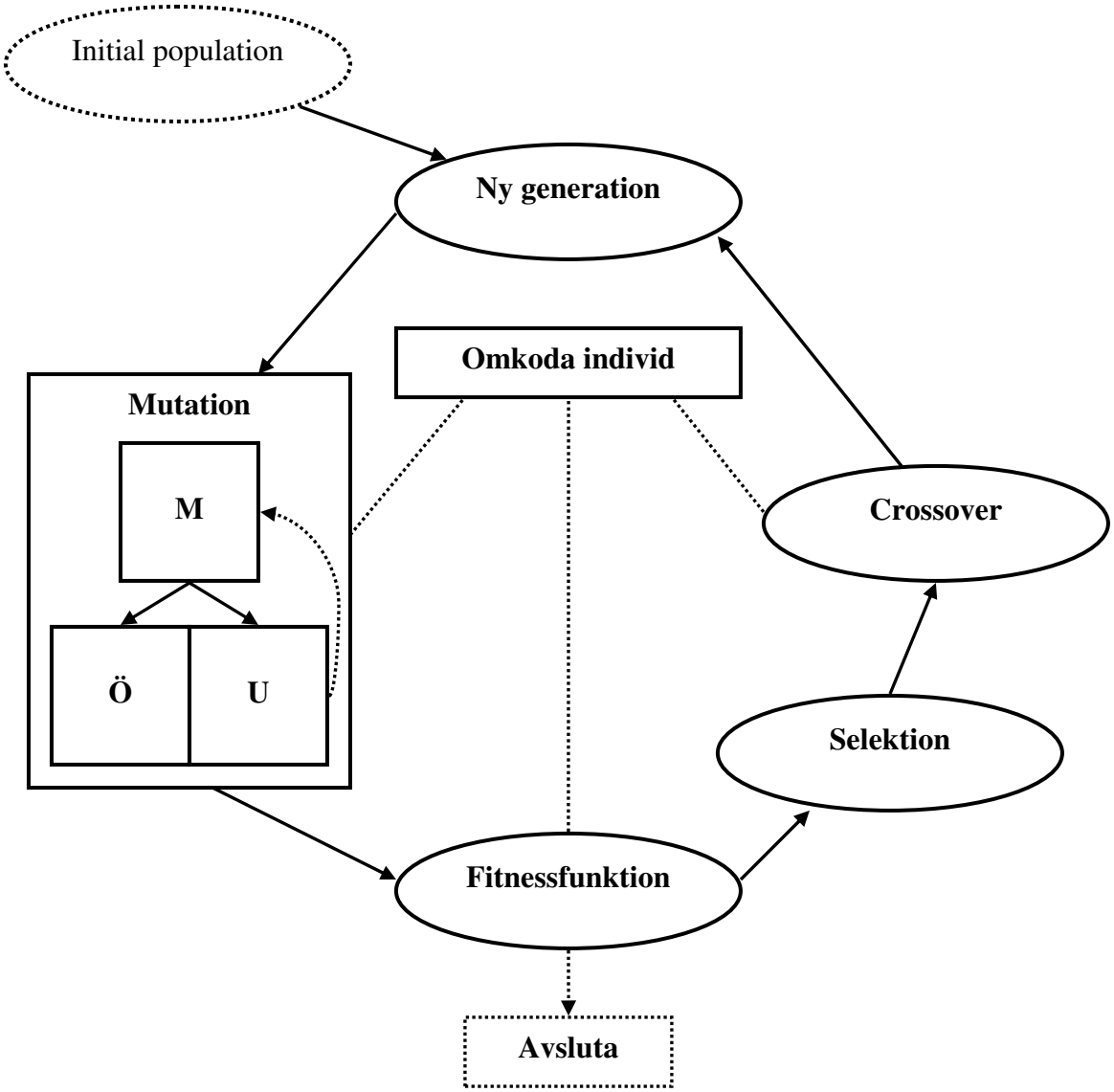
I mutationsoperatorn ingår det alltså tre olika funktioner. En som beräknar överskott eller underskott och två funktioner som behandlar respektive utfall. I fortsättningen går dessa funktioner under beteckningen ”M” för beräkningsfunktionen, ”Ö” och ”U” för överskott respektive underskott.

Det behövs även en funktion för att omkoda en individ från procentuella värden till heltal, utifrån den förväntade försäljningen för respektive produkt.

Hur länge den genetiska algoritmen skall tillåtas arbeta är oklart då den egentligen måste testas för det skall gå att avgöra dess prestanda. Det borde dock rimligt för ett företag att avgöra vad en ungefärlig godtagbar avkastning skulle kunna vara. I följande kapitel skall en modell för en genetisk algoritm utvecklas. Modellen skall hjälpa mig att förklara hur den genetiska algoritmen skall angripa produktionsproblemet och samtidigt stärka mina slutsatser om att en genetisk algoritm skulle fungera mot produktionsproblemet.

I bilaga 1 har jag manuellt genomgått de olika stegen i den utvecklade modellen. Där visar jag med ett exempel hur den genetiska algoritmen är tänkt att fungera mot produktionsproblemet.

5.3 Modell över den genetiska algoritmen



Figur 16: Modell över den föreslagna genetiska algoritmen.

För att förklara den ovanstående konceptuella modellen börjar vi i bubblan som är döpt till ”initial population”. I detta steg skapas den första generationen som helt bygger på slumpade värden. Denna population blir vår första generation och den genetiska algoritmen är redo att börja arbeta. Följer vi pilarnas riktning hamnar vi i bubblan döpt

till ”mutation”. Detta skiljer sig från vad teorin säger, då det tycks vara vanligast att mutationsoperatorn sätts i bruk först efter att en crossover har skett.

Anledningen till att mutationen sker redan nu, är att den delvis har i uppgift att skapa variation bland individer, men även att anpassa individerna och undvika otillåtna individer. I steg ”M” bedöms om funktionen ”Ö” eller ”U” skall sättas i bruk. I båda fallen modiferas individen på ett sådant sätt som har beskrivits i kapitel 5.2. När mutationen är genomförd skickas individen till fitnessfunktionen. Individen måste omkodas, vilket kräver att algoritmen kallar på funktionen döpt till ”omkoda individ”. Individen omkodas då från procentuella värden till ett heltal för respektive produkt. Fitnessfunktionen beräknar en sannolik avkastning och varje individ tilldelas ett fitnessvärde.

Populationen skickas sedan till bubblan döpt till ”selektion”. Här sker ett urval enligt roulettehjulsprincipen där ett antal föräldrar ingår i urvalet för att bilda nästa generation. Antalet föräldrar är helt beroende av populationsstorleken och lämpliga värden går endast att testa sig fram till genom experiment, vilket jag inte avser att göra i denna uppsats.

Det urval som uppkommit genom selektion skickas vidare till bubblan döpt till ”crossover”. Enligt två-punkts crossover skapas variation bland individerna och en ny generation skapas. Den nya generationen skickas till bubblan med namnet ”ny generation” och cykeln upprepas.

Den genetiska algoritmen upprepas tills ett slutvillkor blir uppfyllt vilket testas i fitnessfunktionen. Utgångsvärdet kan vara då ett lämpligt antal generationer har genererats eller då en individ med en godtagbar avkastning har skapats.

6 Analys av arbetet med produktionsproblemet

I detta kapitel skall en kort analys över hur arbetet med produktionsproblemet har gått och vilka upptäcker och erfarenheter arbetet har medfört.

Jag har i arbetet med att utveckla min modell över en genetisk algoritm, till att börja med använt mig av en representation som är speciellt anpassad till det aktuella problemet. Min första tanke var att låta varje cell innehålla ett heltal som enkelt representerade det antalet produkter som skall produceras av den typen. Eftersom detta skulle kunna leda till att en viss produkt producerades i ett större antal än den förväntade försäljningen skulle det bryta mot företagets önskemål. Det skulle alltså innebär att en otillåten individ skulle kunna genereras, vilket inte är önskvärt.

Den valda representationen skiljer sig från vad teorin säger och i överensstämmelse med mina erfarenheter från analysen i kapitel fyra, tyder det på att representationen måste anpassas utifrån det aktuella problemet. Detta är en mycket viktig faktor en utvecklare måste ta hänsyn till vid utveckling av en genetisk algoritm.

I detta fall finns det även andra problem som kan medföra att otillåtna individer genereras. En individ kan lätt genereras på ett sådant sätt att produktionskostnaden överstiger den önskade budgeten och en otillåten individ har uppkommit. För att undvika det nämnda problemet, valde jag att utveckla en mutationsoperator som hanterade det aktuella problemet. Teorin säger att en mutationsoperator är till för att skapa slumpmässig variation, men i den utvecklade algoritmen har jag även använd mutationsoperatorn till att försäkra mig om att inga otillåtna individer uppkommer. Utifrån denna erfarenhet vill jag påstå att utformning av genetiska algoritmer kan variera kraftigt och det är utvecklaren till algoritmen som måste tänka över vilka möjligheter det finns att skapa en så effektiv algoritm som möjligt.

Min valda fitnessfunktion överensstämmer relativt väl med vad teorin säger och jag har alltså använt mig av en traditionell kontroll av individernas fitness. Det samma gäller vid selektionen där jag använt mig av en traditionell roulettehjulsprincip. De individer med högst fitness har större chans att ingå i urvalet, men när roulettehjulet snurrar finns det dock en chans att de individerna med högst fitness inte blir valda.

Utifrån mitt arbete med att utveckla en genetisk algoritm har jag upplevt att genetiska algoritmer bygger på ett visst tankesätt, nämligen att lösningar skall genom en konstgjord evolution utvecklas. Detta låter avancerat, medan det hela egentligen handlar om en mer traditionell algoritm med olika typer av funktioner, direkt anpassade för att kunna lösa det specifika problemet.

För att kunna skapa en väl fungerande genetisk algoritm tror jag det är viktigast att utvecklaren är väl insatt i problemet och har god förståelse för problemets olika delar och uppbyggnad. Utifrån dessa förutsättningar kan en utvecklare ta fram en genetisk algoritm, med funktioner speciellt anpassad till det aktuella problemet. Mina erfarenheter tyder på

att det inte finns några tydliga gränser över hur en viss typ av operator kan se ut eller får se ut. Det är upp till algoritmens utvecklare att bestämma detta och på så sätt kan varje genetisk algoritm bli unik, även om de är framtagna att arbeta mot snarlika problem.

7 Metoddiskussion

Mina erfarenheter av att genomföra min informationsinsamling genom att studera litteratur och genomföra en dokumentstudie är goda. Dels har jag kunnat ägna all min tid åt att koncentrera mig på uppsatsskrivandet och dess olika faser. En annan detalj som underlättat mycket, är att jag inte behövt hålla kontakt med företag och inte behövt anpassa min tid till möten för intervjuer etc. Dessa fördelar är till stor del fördelar jag har upplevt som person, men jag har även sett rent akademiska fördelar. Vid exempelvis intervjuer kan resultatet påverkas av mina ställda frågor och även om jag har möjlighet att ställa följdfrågor, är det lätt hänt att det material jag får med mig till analysen saknar någon viktig del. Genom att studera de skrivna artiklarna, kan jag antaga att författarna har presenterat de allra viktigaste upptäckterna och inte undanhållit viktig information. Mina erfarenheter av den genomförda dokumentstudien är dock goda och jag anser att jag inte hade fått ut mer information genom en intervju eller en fallstudie genomfört på något annat sätt.

Att arbeta mot the traveling salesman under min dokumentstudie upplevde jag som ett bra problemområde. Att studera genetiska algoritmer och deras tillämpningar mot problemet, fick mig att inse många svårigheter och nackdelar med genetiska algoritmer. Det går att spekulera hur vida det hade påverkat denna uppsats om jag hade undersökt hur genetiska algoritmer tillämpas mot andra eller ett blandat antal problemområden. Jag tror inte att the traveling salesman problem har inverkat på mitt resultat, utan de observationer jag gjort hade kommit fram i vilket problemområde jag än valt.

Jag upplever att min litteraturstudie av genetiska algoritmer har gett en bild över genetiska algoritmer och att de studerade tillämpningarna har gett en lite annorlunda bild av genetiska algoritmer. Både dessa sidor har jag upplevt som viktiga för att kunna förstå hur genetiska algoritmer fungerar och viktiga för att kunna presentera och introducera läsaren till genetiska algoritmer.

Min analysmetod utvecklade jag parallellt med att jag utförde min dokumentstudie. Jag tycker att analysmetoden innehåller viktiga frågeställningar och kände ingen saknad av någon frågeställning eller att analysmetoden begränsade mig på något negativt sätt. Det positiva med min analysmetod var att jag mer specifikt observerade vissa frågeställningar, som jag kanske inte annars hade tänkt på.

Jag har arbetat efter en kvalitativ ansats och jag ser fortfarande detta val som den mest lämpliga vägen att gå i denna uppsats. En kvantitativ ansats hade betytt att uppsatsen hade sett väldigt annorlunda ut och jag tror att den hade blivit mycket mer tidskrävande att genomföra. I denna uppsats har jag istället gått djupare in på hur genetiska algoritmer fungerar, dels i teorin och dels i tillämpningar mot the traveling salesman problem. Jag upplever att detta har gett mig tillräckligt med insikt i de genetiska algoritmerna för att kunna dra vissa slutsatser och även utveckla en modell för att behandla ett produktionsproblem.

8 Slutsatser

I detta kapitel skall de upptäckter och slutsatser presenteras, som uppkommit i och med arbetet med denna uppsats. En teoretisk studie har genomförts och därefter har tillämpningar av genetiska algoritmer mot the traveling salesaman studerats. Till sist har ett nytt problemområde identifierats och behandlats med hjälp av genetiska algoritmer, genom att jag har utvecklat en modell över en genetisk algoritm.

En upptäckt i och med denna uppsats har varit att teoretiska genetiska algoritmer ofta följer ett visst mönster, medan verkligheten ser annorlunda ut. Varje fall är unik och på samma sätt får den genetiska algoritmen ett unikt utseende. För att utveckla en genetisk algoritm i syfte att behandla ett problem, måste utvecklaren välja genetiska operatorer som passar till problemet. Det är här svårigheten ligger i att utveckla en genetisk algoritm.

Felaktig representation av individer kan leda till att otillåtna individer skapas, vilket kan ställa till stora problem för den genetiska algoritmen. På liknande sätt kan olämpliga mutationsoperatorer eller crossoveroperatorer skapa individer som inte motsvarar en tillåten lösning. Detta blir avgörande frågor för en utvecklare som skall designa en genetisk algoritm. Vilket jag även fick befara själv under utvecklingen av modellen över den genetiska algoritmen som tagits fram för att lösa produktionsproblemet.

Otillåtna individer tycks vara ett av de större problemen med genetiska algoritmer. I den modell jag utvecklat för produktionsproblemet har jag valt att använda mig av en mutationsoperator som kontrollerar individen och anpassar den, i syfte att undvika otillåtna individer. En utvecklare bör alltid beakta möjligheterna att otillåtna individer kan uppstå och skapa sig en förståelse för vilka konsekvenser detta kan leda till. Otillåtna individer kan både göra mer eller mindre skada.

Teorin och de två dokumentstudierna visar alla på olika typer av selektion, crossoveroperatorer och muatationsoperatorer. Till den genetiska algoritmen, designad för att lösa produktionsproblemet, har jag tagit vara på dessa olika operatorer och det har hjälpt mig vid utveckling och design av modellen. En annan upptäckt är att ordningen på den genetiska operatorns olika delar kan skilja sig. I min modell har jag valt att placera mutationsoperatorn före fitnessfunktionen och selektionen. Detta av den anledningen att det finns ett behov att reglera individerna för att undvika otillåtna lösningar. Detta visar än en gång att problemet styr den genetiska algoritmens utseende.

Av detta kan jag dra slutsatserna att det inte finns några direkta regler över hur en genetisk algoritm skall se ut. Det är konceptet som är viktigt, där individer i en population tävlar mot varandra för att få plats i ett urval. Urvalet kombineras och förändras på något sätt för att generera en ny generation, i syfte att skapa adaption.

Jag kan även konstatera att de två studerade försöken att angripa the traveling salesman problem med hjälp av genetiska algoritmer var relativt lyckade. I den första undersökta artikeln av Tsai H., Yang J., Tsai Y. & Kao C., (2004) hittades den optimala lösningen för problem med under sextusen noder och den genomsnittliga felgränsen blev 0,0074 procent från den optimala lösningen på det störta problemet (13509 noder), som ingick i experimentet. Samtidigt som Harmel B. & Patterson M., (2003) beskriver i sin artikel hur de löser TSP-problem genom linjär programmering i kalkylprogrammet Microsoft Excel och hur problem uppstår redan vid tiotalet noder. Detta tycker jag tyder på att genetiska algoritmer ger ett bra resultat och i jämförelse mellan ovanstående fall, är den genetiska algoritmen mer kraftfull än den lösningen där linjär programmering används i Excel. Denna uppsats har dock inga intensioner att jämföra genetiska algoritmer mot någon annan metod och jag har inte haft till avseende att undersöka om någon metod är bättre än någon annan.

All typ av problemlösning är dock problemspecifik. Det har visat sig att genetiska algoritmer går att använda mot många olika områden. Genetiska algoritmer har tillämpats vid inlärning av styrprogram inom robotik (Nordin P. & Wilde J., 2003), för att lösa the traveling salesman problem (TSP) (Li G. & Louis S., 2000) och vid design av datoriserade nätverks topologier (Legault G. & Pierre S., 1998) och jag har även visat i denna uppsats hur jag angriper ett produktionsproblem med hjälp av genetiska algoritmer. Alla dessa problem har olika uppbyggnad och visar på den bredd av användningsområden där genetiska algoritmer kan vara verksam mot.

Referenslista

Aronson J. & Turban E. (2001) *"Decision Support Systems and Intelligent Systems"*, Prentice-Hall, Inc., Upper Saddle River, New Jersey, ISBN 0-13-032723-9

Blaxter L., Hughes C. & Tight M. (2001) "*How To Research"*, Second Edition, Open University Press, Buckingham, ISBN 0-335-20903-3

Connolly B., 2004 *"Survival of the Fittest: Natural Selection with Windows Forms"*, MSDN Magazine, The Microsoft Journal for Developers, August 2004

Golden B., Pepper J. & Wasil E. (2002) *"Solving the Traveling Salesman Problem With Annealing-Based Heuristics: A Computational Study"*, IEEE Transactions On Systems, Man And Cybernetics - Part A: Systems And Humans, Vol. 32, No. 1, January 2002

Harmel B. & Patterson M. (2003) "*An Algorotithm for Using Excel Solver for the Traveling Salesman Problem",* Journal of Education for Business, July/August 2003

Hwang S. & Kuo T. (1996) *"A Genetic Algoritm with Disruptive Selection"*, IEEE Transactions On Systems, Man and Cybernetics - Part B: Cybernetics, Vol. 26, No. 2, April 1996

Juang C. (2004) *"A Hybrid of Genetic Algorithm and Particle Swarm Optimazation for Recurrent Network Design"*, IEEE Transactions On Systems, Man and Cybernetics - Part B: Cybernetics, Vol. 34, No. 2, April 2004

Legault G. & Pierre S. (1998) *"A Genetic Algorithm for Designing Distributed Computer Network Topologies"*, IEEE Transactions On Systems, Man and Cybernetics - Part B: Cybernetics, Vol. 28, No. 2, April 1998

Leou J. & Lin D. (1997) "*A genetic algoritm approach to chinese handwriting normalization*", IEEE Transactions On Systems, Man and Cybernetics - Part B: Cybernetics, Vol. 27, No. 6, December 1997

Li G. & Louis S. (2000) *"Case injected genetic algorithms for traveling salesman problems"*, Information Science, Vol. 122, 2000

Lundh L., Montgomery H. & Waern Y. (1992) *"Kognitiv psykologi"*, Studentlitteratur, Lund, ISBN 91-44-35931-4

Lundin R. (2003) *"Identitet och beslutsfattande – konstitution, relation och kreation"*, Pedagogiska institutionen, Stockholm Universitet, Stockholm, ISBN 91-628-5590-5

Marakas G. (2003) *"Decision support systems – In the 21st Century"*, Pearson Education, Inc., Upper Sadddle River, New Jersey, ISBN 0-13-092206-4

Nordin P. & Wilde J. (2003) *"Humanoider - Självlärande robotar och artificiell intelligens"*, Liber AB, Stockholm, ISBN 91-47-05191-4

Olsson B. (1996) *"Genetiska Algoritmer – Introduktionskompendium, Sommarkurs 1996"*, Department of Computer Science, University of Skövde, Skövde, 1996

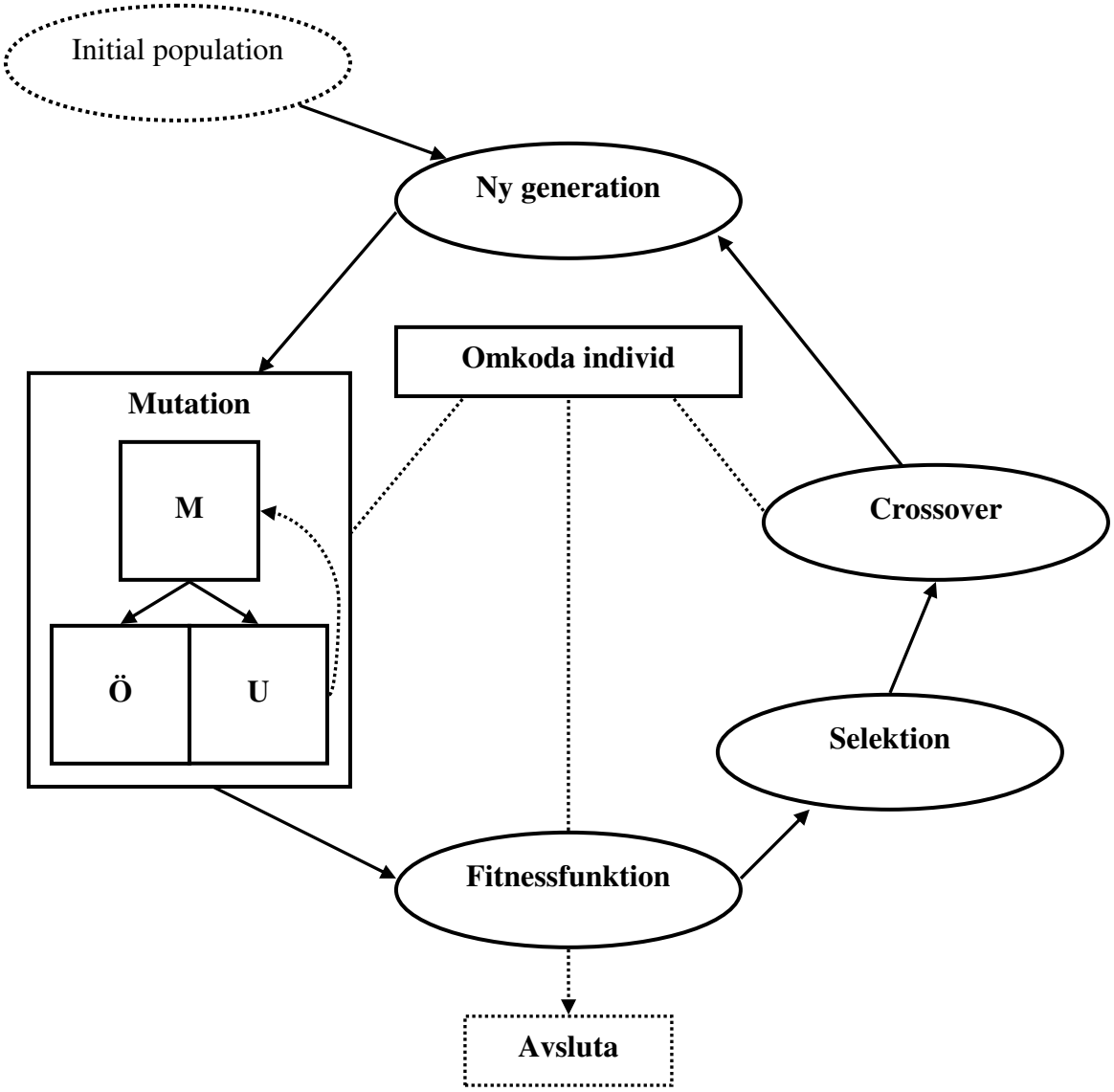
Pearl J. (1988) *"Probabilistic Reasoning in Intelligent Systems"*, Morgan Kaufmann Publishers, Inc, San Fransisco, CA, ISBN 1-55860-479-0

Salem O. & Shahin A. (2004) *"Using genetic algorithms in solving the one-dimensional cutting stock problem in the construction industry"*, Canadian Journal of Civil Engineering, Apr 2004, 31, 2, 2004

Tsai H., Yang J., Tsai Y. & Kao C. (2004) *"An Evolutionary Algorithm for Large Traveling Salesman Problems"*, IEEE Transactions On Systems, Man and Cybernetics - Part B: Cybernetics, Vol. 34, No. 4, August 2004

Yin R. (2003) *"Case study research – design and methods"*, Sage Publications, Thousand Oaks, CA, ISBN 0-7619-2552

Bilaga 1



Ovan finns modellen över den genetiska algoritm som är designad att lösa mitt valda produktionsproblem. I denna bilaga vill jag genom ett enkelt exempel visa hur den genetiska algoritmen skulle kunna arbeta mot ett specifikt problem. I exemplet finns det sex stycken olika produkter, som beskrivs i nedanstående tabell. Raden ”försäljning” står för den förväntade försäljningen, medan ”kostnad” står för kostnaden att producera en enhet av den aktuella produkten och på samma sätt står ”pris” för det aktuella försäljningspriset för en enhet av den aktuella produkten.

Produktlista

Produkt:	Produkt 1	Produkt 2	Produkt 3	Produkt 4	Produkt 5	Produkt 6
Försäljning:	10	20	15	10	10	12

Kostnad:	100	80	90	40	10	50
Pris:	150	150	200	80	30	80

Företaget har i sin budget bestämt att den totala produktionskostnaden inte får överstiga 2000 kronor. De har satt två slutvillkor som bestämmer när den genetiska algoritmen skall sluta. Det första villkoret är att en godtagbar lösning ger en vinst på 2300 kronor, vilket medför att om en skapad individ ger en vinst på 2300 kronor eller mer så skall den genetiska algoritmen stanna och presentera den aktuella lösningen. Det andra avslutningsvillkoret är att den genetiska algoritmen skall stanna efter 20 generationer om inte en godtagbar lösning har hittats.

1.1 En ny generation

En initial population genereras slumpmässigt och består i det här fallet endast av fyra individer. Den nya generationen hittas i den översta bubblan i modellen och den genetiska algoritmen kan börja sitt arbete.

Ny generation

100	20	40	30	20	50
20	10	40	50	100	50
70	10	40	100	20	70
10	20	40	30	20	10

1.2 Mutation

Funktionen M beräknas mot initialpopulationen, i syfte att bestämma om funktion Ö eller U skall aktiveras. Först anropas funktionen för att omkoda individerna och skapar en vektor för respektive individ, där siffrorna står för antalet produkter som produceras, av respektive produkt.

Generation omkodad till numeriska värden

10	4	6	3	2	6
2	2	6	5	10	6
7	2	6	10	2	8
1	4	6	3	2	1

Därefter beräknar funktion M ut om individen motsvarar ett överskott eller underskott.

Individernas respektive produktionskostnad

1000	320	540	120	20	300	=2300
200	160	540	200	100	300	=1500
700	160	540	400	20	400	=2220
100	320	540	120	20	300	=1400

Budgeten är sedan tidigare satt till 2000 kronor och individ nummer 1 och 3 lider av ett underskott, medan individ nummer 2 och 4 lider av ett överskott, enligt sista rutan i ovanstående matris.

Här näst visar jag hur funktionen U hanterar individ nummer 1 och 3 och hur funktionen Ö hanterar individ nummer 2 och 4.

Individ nummer 1 (oförändrad)

10	4	6	3	2	6

Underskott (kr): 2300 – 2000 = 300

Därefter slumpas en aktuell cell i vektorn:

10	4	6	3	2	6

Underskottet divideras med kostnaden för en producerad enhet av den valda produkten.

300 / 90 = 3,33

Resultatet avrundas uppåt till närmaste heltal, vilket är ”4”. Funktionen U tar därefter bort 4 stycken enheter från den aktuella cellen och modifierar individen på följande sätt:

Individ nummer 1 (förändrad)

10	4	2	3	2	6

Enligt samma princip modifieras individ nummer 3 och får följande utseende:

Individ nummer 3 (förändrad)

7	2	6	10	2	3

Individ nummer 2 och 4 skall nu bearbetas med funktionen Ö som hanterar överskott i en individ.

Individ nummer 2 (oförändrad)

2	2	6	5	10	6

Överskott: 2000 - 1500 = 500

Därefter slumpas en aktuell cell i vektorn:

2	2	6	5	10	6

Överskottet divideras med kostnaden för en producerad enhet av den valda produkten.

500 / 100 = 5

Det finns inget behov av att avrunda resultatet i det här fallet, utan funktionen Ö lägger därefter till 5 enheter i den aktuella cellen.

Individ nummer 2 (förändrad)

7	2	6	5	10	6

Individ nummer 4 modifieras enligt samma principer och får följande utseende:

Individ nummer 4 (förändrad)

6	4	6	3	2	1

1.3 Fitnessfunktionen

När alla individer i populationen har modifierats till att vara godkända individer skall evalueringen ske. Den totala avkastningen för respektive individ beräknas genom att beräkna den totala försäljningen och subtrahera den totala produktionskostnaden. Resultatet presenteras i den sista cellen på respektive rad.

Den totala vinsten av respektive individ

10	4	2	3	2	6	1340
7	2	6	5	10	6	1730
7	2	6	10	2	3	1680
6	4	6	3	2	1	1480

1.4 Selektion

För att genomföra selektion enligt roulettehjulsprincipen skall individernas procentuella fördelning beräknas.

Totalt: 6230
Individ 1: 1340 / 6230 = 0,21 → 21%
Individ 2: 1730 / 6230 = 0,27 → 27%
Individ 3: 1680 / 6230 = 0,27 → 27%
Individ 4: 1480 / 6230 = 0,24 → 24%

Då roulettehjulet snurras sker ett urval där två individer överlever och de två andra lösningarna förkastas. Urvalet i det här fallet föll på individ nummer 2 och 3.

1.5 Crossover

Dessa två individer är de som skall utsättas för den genetiska algoritmens crossoveroperator:

Individerna 2 och 3 angivna i procentuella värden

70	10	40	50	100	50
70	10	40	100	20	25

De två individerna utsätts för två-punkts crossover två gånger för att generera fyra stycken nya individer, som kan ersätta den förra generationen. Två-punkts crossover utförs genom att två crossoverpunkter väljs slumpmässigt och innehållet mellan punkterna byter plats.

Crossover 1
De slumpmässigt valda crossoverpunkterna blir position 2 och 4, vilket leder till nedanstående förändring.

Före crossover

70	10	40	50	100	50
70	10	40	100	20	25

Efter crossover

70	10	40	100	100	50
70	10	40	50	20	25

Crossover 2
De slumpmässigt valda crossoverpunkterna blir position 3 och 6, vilket leder till nedanstående förändring.

Före crossover

70	10	40	50	100	50
70	10	40	100	20	25

Efter crossover

70	10	40	100	20	25
70	10	40	50	100	100

En ny population har nu genererats. I vanliga fall arbetar den genetiska algoritmen med större populationer och redan här går det att se att variationen inom populationen inte är speciellt hög, vilket är ett direkt resultat av att populationen endast bygger på två individer från föregående population. Den nya populationen ser alltså ut enligt följande:

Ny population

70	10	40	100	100	50
70	10	40	50	20	25
70	10	40	100	20	25
70	10	40	50	100	100

Den nya populationen ersätter den föregående populationen och på samma sätt genomgår denna population mutation, fitnessfunktionen, selektion och crossover, för att återigen skapa en ny population. Den genetiska algoritmen fortsätter sitt arbete tills ett avslutningsvillkor har uppfyllts.