ISSN 0280–5316
ISRN LUTFD2/TFRT--5485--SE

Produktionsplanering

med hjälp av Genetiska Algoritmer och Neurala Nätverk

Ola Johansson
Magnus A R Andersson

Institutionen för Reglerteknik
Lunds Tekniska Högskola
Oktober 1993

Department of Automatic Control
Lund Institute of Technology
P.O. Box 118
S-221 00 Lund Sweden

Document name
MASTER THESIS

Date of issue
October 1993

Document Number
ISRN LUTFD2/TFRT--5485--SE

Author(s)
Ola Johansson and Magnus A R Andersson

Supervisor
R. Johansson, H. Olsson, I. Kronfeldt, N-O Jönsson

Sponsoring organisation
Sydkraft AB, Malmö

Title and subtitle
Produktionsplanering med hjälp av Genetiska Algoritmer och Neurala Nätverk. (Production Planning using Genetic Algorithms and Neural Networks).

Abstract

The main problem for Sydkraft, and all other power suppliers, is to provide their customers with the exact amount of electrical power they demand, as economical as possible. This can be divided into two subproblems, to forecast the demand, and to plan the production accordingly. In our Master's thesis we try to solve the last of the here above mentioned subproblems, or how Sydkraft ought to plan operation of their power plants to achieve maximum profit.

The purpose of this Master's thesis has been to determine whether *genetic algorithms* (GA) can be used to optimize the production planning, or not. We have also applied *linear programming* to a linear model over a power system for a comparison. We have shown that GA converges to minimum cost for the production.

We also applied GA to a non-linear model of Sydkraft with the purpose to examine if it can be used in the real world. Our analysis shows that GA tries to minimize the production cost, but the model's complexity and perturbed cost curves results in a failure to complete the optimization within reasonable time.

Key words
Genetic Algorithms, Neural Network, Production planning

Classification system and/or index terms (if any)

Supplementary bibliographical information

ISSN and key title
0280–5316

ISBN

Language
Swedish

Number of pages
58

Recipient's notes

Security classification

INLEDNING

Denna rapport är resultatet av ett examensarbete som utförts huvudsakligen vid institutionen för reglerteknik, Lunds Tekniska Högskola. Arbetet med insamling av data och modellbyggande har dock utförts på Sydkraft AB, huvudkontoret, Malmö.

BAKGRUND

Det primära problemet, för Sydkraft och andra elproducenter är att tillgodose sina kunder med exakt så mycket elkraft de efterfrågar, så ekonomiskt som möjligt. Detta kan delas upp i två delproblem, dels att prognosticera efterfrågan så bra som möjligt och dels att utifrån denna bestämma hur produktionen skall ske så billigt som möjligt. Vårt examensarbete behandlar det sistnämnda av dessa två delproblem, d v s att ta fram planer för hur Sydkraft bör köra sina kraftverk och hur de bör köpa och sälja kraft för att maximera sin vinst.

SYFTE OCH RESULTAT

Syftet med examensarbetet har varit att undersöka huruvida Genetiska Algoritmer kan användas för att ta fram optimala produktionsplaner. För att kunna lösa detta problem har vi modellerat de olika kraftproducerande objektens kostnadsfunktioner. I fallet med handel på kraftbörsen har vi modellerat utbytespriserna med hjälp av neurala nätverk.

Vi har förutom Genetiska Algoritmer [GA], även använt oss av Linjär Programmering [LP] för att minimera produktionskostnaden. Jämförelsen mellan GA och LP på ett linjärt kraftsystem visar att GA utan tvekan konvergerar mot optimum men att GA inte når ända fram. Detta är inget oväntat resultat då GA sällan eller aldrig finner det absoluta optimat.

Vi presenterar även en GA-lösning av ett olinjärt kraftsystem. Vår analys av resultatet visar att GA även här konvergerar mot optimum. Det tar dock väldigt långt tid, även på vår nedskalade modell. Detta beror på att problemet har vissa karakteristiska egenskaper som GA har svårigheter med, däribland kromosomberoende och kostnadsfunktionens utseende nära optimum. Detta innebär att problemet, p g a sin storlek och betydande komplexitet vid en uppskalning av modellen, resulterar i att GA, liksom många andra optimeringsmetoder, misslyckas med att lösa problemet inom en rimlig tid.

TILLERKÄNNANDE

Slutligen vill vi tacka alla de människor som bidragit med hjälp under arbetets gång, framförallt till vår handledare på institutionen för reglerteknik, Rolf Johansson, samt våra handledare på Sydkraft; Hans Olsson, Ingemar Kronfeldt och Nils-Olof Jönsson, för deras osvikliga stöd i med- och motgång. Ett stort tack även till chefen på PDD, Mårten Eriksson, med vars kunskap och hjälp en bra Sydkraft-modell kunde byggas.

Lund, den 5:e Oktober, 1993.

Ola Johansson
Magnus A R Andersson

INNEHÅLL

1 GENETISKA ALGORITMER

I detta inledande kapitel kommer vi att introducera genetiska algoritmer, i fortsättningen kallat GA, för läsaren. Vi kommer att beskriva de grundläggande funktionerna, hur de implementeras på en dator och slutligen exemplifiera GAs beteende, med hjälp av ett par enkla problem.

1.1 INLEDNING

Genetiska algoritmer är en mycket kraftfull optimeringsmetod som skiljer sig väsentligt från de flesta traditionella och mer matematiskt orienterade optimeringsmetoder. Istället för att baseras på derivator och liknande matematiska begrepp, baseras de på Darwins tes "den starkes överlevnad", d v s på evolution och det naturliga urvalet.

Mekaniken bakom evolutionsläran är inte känd i alla detaljer, men vissa karakteristiska drag är kända. Evolutionen äger rum i kromosomerna. Kromosomerna består av en DNA-kedja som innehåller kod för varje detalj i en individs uppbyggnad. Ett levande väsen skapas genom en avkodning av kromosomen. Detaljerna kring kodning och avkodning är inte fullständigt kända men vissa generella drag anses gälla.

1. Evolutionen är en process som arbetar på kromosomer snarare än de levande väsen kromosomen avkodat.
2. Det naturliga urvalet är länken mellan en kromosom och den avkodade individens anpassning till sin miljö. Det naturliga urvalet försäkrar att en väl anpassad individ har större chans att fortplanta sig än en dåligt anpassad.
3. Reproduktionsprocessen är den process där evolutionen sker. Mutationer kan orsaka att avkomman skiljer sig från sina föräldrar. Även den stora mängd kombinationsmöjligheter som finns mellan föräldrarnas kromosomer kan skapa barn som är olika sina föräldrar.
4. Biologisk evolution saknar minne. All kunskap om produktion av väl anpassade individer finns i den nu levande genbanken, d v s de kromosomer som finns i dagens levande väsen.

Genetiska algoritmer har sina rötter i den artificiella intelligensen och forskningen kring tekniker som ger en adaptiv anpassning till skiftande miljöer. Grunden till genetiska algoritmer las i *Adaptation in Natural and Artificial Systems*[1] och författaren John Holland betraktas som skaparen av detta forskningsfält. Dock hade även tidigare datorsimuleringar med genetisk karaktär utförts[2] .

Idén är att man har en population av olika individer, som alla är en tänkbar lösning på problemet man försöker lösa. Varje individ, eller lösning, byggs upp med hjälp av en kromosom som mall. Kromosomen består i sin tur av en kedja med gener, där varje gen innehåller en regel som påverkar individens beteende. Till varje individ hör ett entydigt bestämt tal som är en funktion av kromosomens gener. Detta tal ger individens värde som lösning på det ställda problemet. Detta kan till exempel vara en intäkt som man vill optimera.

Genom att kombinera de starkaste individernas överlevnad med slumpmässiga korsningar av gamla och väl anpassade individer söker man sig mot den optimala lösningen. Varje ny generation skapas genom att använda de bästa individerna av den

---

[1]Holland John, Adaptation in Natural and Artificial Systems, The University of Michigan Press, 1975.

[2]Fraser A S, Simulation of genetic systems, Journal of Theoretical Biology, 2, p 329ff, 1962

Bremermann H J, Optimization through evolution and recombination, In M C Yovits, G T Jacobi & G D Goldstein (eds.), Self-organizing systems, Spartan Books, 1962.

föregående generationen och emellanåt också slumpmässiga nya individer. Även om mycket styrs av slumpen är genetiska algoritmer ingen slumpvandringsalgoritm utan bygger på ett effektivt utnyttjande av historisk information för att hitta nya sökpunkter i lösningsrummet, med bättre uppförande.

Det finns två mekanismer som är grundläggande för att genetiska algoritmer skall fungera. Den ena sorterar bort de individer som inte är anpassade för den miljön de befinner sig i och låter de övriga överleva. Detta naturliga urval sorterar uppenbarligen endast bort dåliga exemplar och räcker inte för att någon utveckling av individerna skall ske. Det är här de genetiska operatorerna kommer in.

Den andra är alltså de genetiska operatorerna som verkar på en eller ett par kromosomer. Ur populationen väljer man ut de individer som tillsammans skall skapa nästa generation. Chansen att en individ skall bli utvald för att skapa nästa generation beror på hur lämplig individen är för sin uppgift, d v s hur bra lösning på problemet individen utgör. Detta är naturligtvis den artificiella versionen av Darwins "Den starkes överlevnad". Individernas kromosomer kopieras och de genetiska operatorerna tillåts verka på detta urval av kromosomer för att utveckla bättre individer.

1.2 IMPLEMENTERING AV GA PÅ DATOR

Genetiska algoritmer är egentligen ett naturligt fenomen som man skapar en modell av. Modellen implementerar man sedan på en dator för att kunna lösa olika typer av problem. Den artificiella representationen av kromosomen är likadan som i naturen, den består av en serie gener. Skillnaden uppkommer vid modelleringen av generna som i datorn består av binära siffror. Vanligast förekommande är att en gen motsvaras av en binär siffra. Detta är också den genrepresentation vi använt oss av.

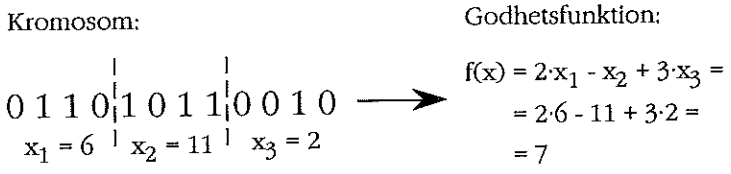
En översiktsmodell över vår första implementering ser ut som nedan.

1. Skapa en slumpmässig initial population av kromosomer.
2. Utvärdera varje kromosom i populationen.
3. Skapa en ny generation genom att para ihop lämpliga kromosomer.
4. Mutera individerna i den nya generationen och flytta runt dem mellan olika delpopulationer. Låt slutligen den nya generationen ta den gamla populationens plats.
5. Gå till punkt 2 tills man nått ett tillfredställande resultat.

Vi ska i de följande avsnitten förklara de olika stegen lite mer detaljerat.

1.2.1 UTVÄRDERING AV KROMOSOMER

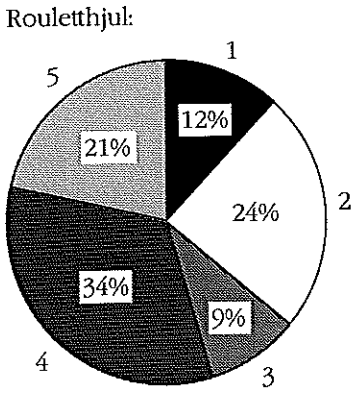
Vid utvärderingen av kromosomerna räknar man ut deras s k godhetstal. Godhetstalet är värdet på kromosomen som lösning på det ställda problemet. Exempelvis kan kromosomen bestå av tre 4-bitars binärkodade heltal, d v s en kromosom med $3 \cdot 4 = 12$ gener. Godhetsvärdet är då lika med värdet av den tredimensionella funktion man vill optimera. Se figur 1.1 nedan för ett exempel hur beräkningen kan gå till.



Figur 1.1 Exempel på beräkning av en kromosoms godhetstal.

1.2.2 URVAL AV FÖRÄLDRAR

Syftet med att använda en urvalsfunktion är att ge de bästa individerna en större chans att fortplanta sig. En av de vanligaste teknikerna är det s k rouletthjulsurvalet som bygger på att man skapar ett rouletthjul där varje individ får ett fack. Storleken på facket eller sektorn är beroende av individens godhetstal. Se figur 1.2 nedan för en illustration av rouletthjulstekniken. Genom att snurra rouletthjulet får man sedan fram kandidater till fortplantningen. De bäst anpassade individerna tenderar då att producera flest avkommor.



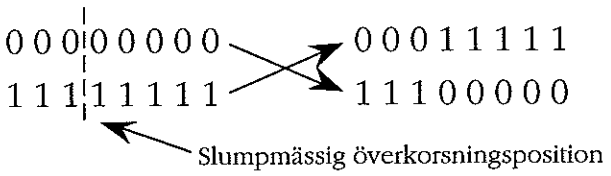
Godhetsvärden:

Individ 1:	4
Individ 2:	8
Individ 3:	3
Individ 4:	11
Individ 5:	7

Figur 1.2 Exempel på rouletthjul för en fem-individers population.

1.2.3 ÖVERKORSNING

Överkorsning är den datalogiska motsvarigheten till könslig fortplantning. I naturen sker detta genom att två kromosomer av samma sort byter gener med varandra. Detta är den främsta genetiska operatorn. Dess effektivitet inses lätt om man tänker sig att man korsar två individer med varsin god egenskap. Om man har tur vid korsningen så hamnar båda dessa goda egenskaper i den ena av de nya kromosomerna som därigenom klarar det naturliga urvalet och dessutom är överlägsen sina föräldrar. Eftersom denna individ med stor sannolikhet är överlägsen de andra individerna i populationen finns det en betydande chans att denna individs kromosom väljs ut när nästa generation skall skapas. På detta sätt sprids det framgångsrika genparet i populationen. I figur 1.3 visas hur överkorsningsoperatorn verkar.



Figur 1.3 Överkorsningsoperatorn

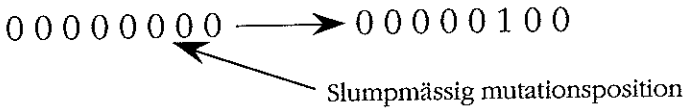
Det finns även mer komplicerade överkorsningsoperatorer som vi kommer att förklara i kapitel 3 i samband med att vi beskriver olika GA-inställningar och GA-funktioner.

1.2.4 MUTATION

Mutation är den genetiska operator som skall se till att evolutionen inte urartar. Detta påstående kräver naturligtvis en förklaring men först en beskrivning av operatorns verkan. En gen väljs ut slumpmässigt och byts ut mot en annan gen. Den nya genen väljs ur mängden möjliga gener och inte bland de gener som finns i populationen. Detta försäkrar oss om att en bra gen som råkat försvinna ur populationen återintroduceras

med en viss sannolikhet. Precis som i naturen har man en relativt låg sannolikhet för mutationer, detta p g a att en hög mutationssannolikhet stör algoritmen mer än den tillför. Kromosomerna i en population där naturligt urval fått verka ett tag är relativt bra anpassade och det är endast små förändringar som kan förbättra dem.

Med en mutationssannolikhet över 0.5 har också algoritmen reducerats till en ren slumpvandringsalgoritm. Figur 1.4 visar mutationsoperatorns funktion. Med binära kromosomer kan mutationsoperatorn beskrivas som en icke-operation på en slumpmässigt vald gen i kromosomen.



Figur 1.4 Mutationsoperatorn

1.2.5 MIGRATION

Populationen är den mängd individer som utgör en generation. Det är viktigt med en lagom stor population, är den för liten är det för liten varians mellan kromosomerna för att nya och bättre kromosomer skall kunna utvecklas. Är populationen istället för stor tar det lång tid innan en förbättring av en gen sprider sig och får genomslag i populationen. Undersökningar[3] har visat att det kan vara fördelaktigt att dela upp populationen i flera halvisolerade kolonier med endast en mycket restriktiv migration mellan dessa. På detta sätt letar sig kolonierna mot optimum på olika vägar och sannolikheten att algoritmen går snett minskar drastiskt. Migrationsfunktionen är den funktion som flyttar runt individerna mellan delpopulationerna.

1.3 GA FÖR OPTIMERING

Genetiska algoritmer är mycket kraftfulla heuristiska sökalgoritmer som använder biologisk evolution som en ram för sina sökprocess. En mycket stor fördel med genetiska algoritmer applicerade på optimeringsproblem är deras stora robusthet och förmåga att undvika att fastna i lokala max utan istället hitta det globala optimumet. En annan stor fördel är också att genetiska algoritmer inte kräver kontinuerliga funktioner, kända derivator etc. Bland nackdelarna hör att man sannolikt inte hittar det optimala värdet, i gengäld hittar man med mycket stor sannolikhet ett väldigt bra värde, d v s mycket nära det optimala

Skillnader mellan genetiska algoritmer och traditionella optimeringsmetoder[4]

1. Genetiska algoritmer arbetar med en kod av parametrarna, inte med parametrarna själva.
2. Genetiska algoritmer söker sig fram med hjälp av en population av sökpunkter och inte med en enstaka punkt.
3. Genetiska algoritmer använder sig av en enkel avkastningsfunktion, inte av derivator eller annan härledd information.
4. Genetiska algoritmer styrs av slumpen inte av deterministiska regler.

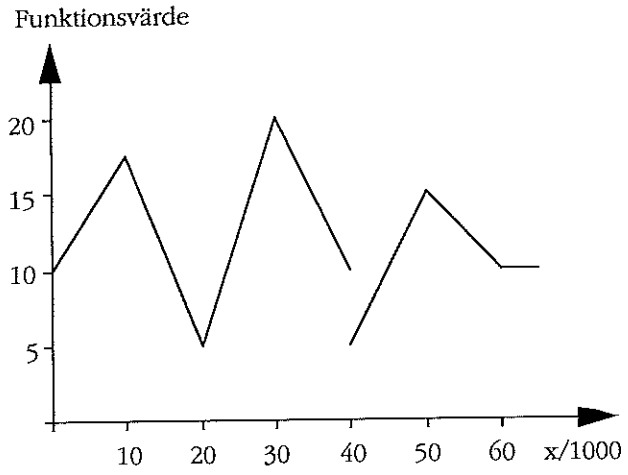
---

[3]Sumida B H m fl, Genetic Algorithms and Evolution, Journal of Theoretical Biology (1990) 147, p59-84, Academic Press Limited, 1990.
[4]Goldberg David E., Genetic Algorithms in Search, Optimization & Machine Learning, Addison Wesley, 1989.

Att genetiska algoritmer arbetar med en kod av parametrarna leder till att man måste diskretisera kontinuerliga funktioner. Detta är dock inte till någon stor nackdel utan kan ofta lösas enkelt.

1.3.1 EXEMPEL 1, EN ENDIMENSIONELL FUNKTION

För att exemplifiera genetiska algoritmers fördelar respektive nackdelar har vi testat dem på ett enkelt exempel. Vi har valt en 16-bitars kromosom som är ett direkt kodat binärt tal, vi har alltså ett möjligt talområde från 0 till 65535. Godhetsfunktionen ser ut som figur 1.5 visar:



Figur 1.5 Avkastningsfunktion

Vi ser att funktionen både har flera lokala max och är diskontinuerlig. En körning med genetiska algoritmer ger följande resultat:

```
Genetic parameters:
-------------------------------------------------------------------------------
Population size:                        10
Maximum # of generations:               10
Chromosome length:                      16
Crossover probability:                  0.60
Mutation probability:                   0.030
-------------------------------------------------------------------------------
Gen: 0   Max: 17.93  Min:  5.13  Average: 12.66  Ncrossover:  0  Nmutation:  0
Gen: 1   Max: 17.93  Min: 11.07  Average: 14.23  Ncrossover:  2  Nmutation:  9
Gen: 2   Max: 19.66  Min: 11.78  Average: 16.59  Ncrossover:  3  Nmutation: 12
Gen: 3   Max: 19.93  Min: 10.00  Average: 16.55  Ncrossover:  6  Nmutation: 17
Gen: 4   Max: 19.93  Min: 10.00  Average: 15.87  Ncrossover:  9  Nmutation: 21
Gen: 5   Max: 19.93  Min: 12.39  Average: 18.15  Ncrossover: 10  Nmutation: 24
Gen: 6   Max: 19.93  Min: 10.00  Average: 17.64  Ncrossover: 13  Nmutation: 28
Gen: 7   Max: 19.98  Min: 11.85  Average: 18.46  Ncrossover: 17  Nmutation: 36
Gen: 8   Max: 19.95  Min:  7.74  Average: 16.86  Ncrossover: 22  Nmutation: 43
Gen: 9   Max: 19.95  Min: 10.00  Average: 16.32  Ncrossover: 24  Nmutation: 51
Gen: 10  Max: 19.95  Min: 13.43  Average: 17.79  Ncrossover: 26  Nmutation: 56
-------------------------------------------------------------------------------
Best individual: 19.98  Chromosome: 0111010101000100,  x: 30020
```

Vi ser att maxvärdet vid denna körningen är 19.98 och detta max hittas i generation sju. Endast 0.051% av det giltiga talområdet har ett funktionsvärde som överskrider det funna maxvärdet. Detta exempel visar styrkan med genetiska algoritmer, d v s de optimerar direkt på avkastningskurvan utan omvägar via derivator etc. Funktionskurvan får också ha flera lokala max och diskontinuiteter. Till svagheterna hör dock att man sällan eller aldrig hittar det absoluta maxvärdet, men man kommer alltid väldigt nära.

1.3.2 EXEMPEL 2, EN FLER-DIMENSIONELL FUNKTION

Vi har också testat ett lite mera komplicerat problem, nämligen:

Maximera: 200*A + 120*B + 160*C

Under bivillkoren:

6*A	+4*B	+5*C	<= 600
A	+B	+C	<= 110
A			<= 50
B			>= 20

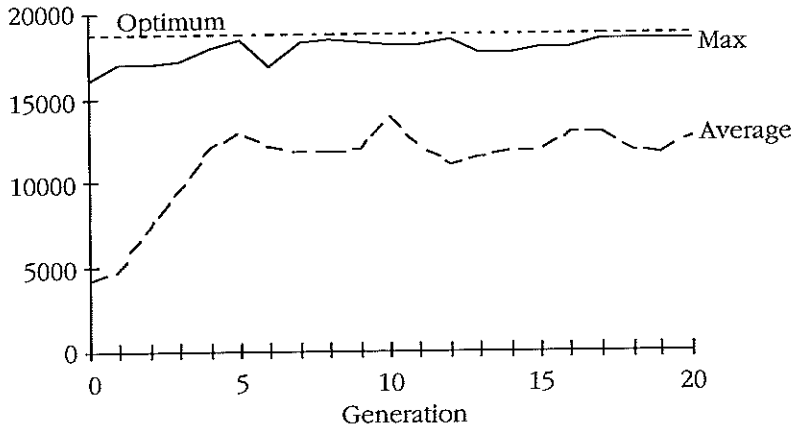
Detta är en problemtyp som löses mycket enkelt med linjär programmering. Optimum finner man i punkten (A,B,C) = (50,20,40), och funktionens värde blir där 18800. Att lösa detta problem med hjälp av genetiska algoritmer är betydligt knepigare. Vi började med att koda varje variabel som ett 8 bitars binärt tal, d v s i intervallet 0-255. De tre variablerna ger då en 24 bitar lång kromosom. Några körningar visade att intervallet inte behövde vara så stort utan kunde minskas från 8 till 6 bitar. Denna minskning hade stor betydelse då de extra bitarna störde algoritmen genom att skapa otillåtna lösningar, d v s bivillkoren överträddes.

Det är dock inte så att otillåtna lösningar är något ovanligt eller ens oönskat, även de otillåtna kromosomerna kan ha genetiskt material av stor vikt. Väsentligt är dock att kodningen av kromosomen inte genererar onödigt många av dessa otillåtna individer, då de tar över populationen i kraft av sitt stora antal. Överträdelserna straffar man genom någon slags straffunktion, i vårt fall använde vi oss av ett straff som var proportionellt mot överträdelsens storlek. Nedan visar vi en körning med den kortare kromosomlängden:

```
Genetic parameters:
-------------------------------------------------------------------------------
Population size:                    50
Maximum # of generations:           20
Chromosome length:                  18
Crossover probability:              0.60
Mutation probability:               0.030
-------------------------------------------------------------------------------
Gen: 0   Max: 16040.00  Min: 0.00  Average: 4258.40   Ncross: 0    Nmut: 0
Gen: 1   Max: 17040.00  Min: 0.00  Average: 4856.80   Ncross: 15   Nmut: 30
Gen: 2   Max: 17040.00  Min: 0.00  Average: 7482.80   Ncross: 30   Nmut: 51
Gen: 3   Max: 17240.00  Min: 0.00  Average: 9639.20   Ncross: 44   Nmut: 72
Gen: 4   Max: 18040.00  Min: 0.00  Average: 12160.40  Ncross: 55   Nmut: 103
Gen: 5   Max: 18440.00  Min: 0.00  Average: 12922.00  Ncross: 72   Nmut: 130
Gen: 6   Max: 17000.00  Min: 0.00  Average: 12142.80  Ncross: 86   Nmut: 159
Gen: 7   Max: 18200.00  Min: 0.00  Average: 11808.80  Ncross: 98   Nmut: 186
Gen: 8   Max: 18400.00  Min: 0.00  Average: 11782.00  Ncross: 114  Nmut: 224
Gen: 9   Max: 18280.00  Min: 0.00  Average: 12056.40  Ncross: 131  Nmut: 254
Gen: 10  Max: 18160.00  Min: 0.00  Average: 13846.00  Ncross: 146  Nmut: 272
Gen: 11  Max: 18160.00  Min: 0.00  Average: 12170.80  Ncross: 159  Nmut: 298
Gen: 12  Max: 18480.00  Min: 0.00  Average: 11056.40  Ncross: 176  Nmut: 335
Gen: 13  Max: 17760.00  Min: 0.00  Average: 11492.40  Ncross: 194  Nmut: 360
Gen: 14  Max: 17560.00  Min: 0.00  Average: 11871.20  Ncross: 211  Nmut: 391
Gen: 15  Max: 18000.00  Min: 0.00  Average: 11879.60  Ncross: 226  Nmut: 415
Gen: 16  Max: 18000.00  Min: 0.00  Average: 12863.60  Ncross: 241  Nmut: 439
Gen: 17  Max: 18520.00  Min: 0.00  Average: 12874.80  Ncross: 259  Nmut: 462
Gen: 18  Max: 18360.00  Min: 0.00  Average: 11871.60  Ncross: 275  Nmut: 487
Gen: 19  Max: 18360.00  Min: 0.00  Average: 11608.40  Ncross: 293  Nmut: 520
Gen: 20  Max: 18360.00  Min: 0.00  Average: 12635.60  Ncross: 306  Nmut: 551
-------------------------------------------------------------------------------
Best individual: 18520.00  Chromosome: 110101001010111101
```

Bästa värdet, 18520, fick vi i generation 17. Detta värde avviker endast 1.5% från det optimala. Lösningen blev (A,B,C)=(43,20,47). Vi tittar lite närmre på hur populationens maxvärde och genomsnittsvärde varierar i figur 1.6 nedan. Vi ser att maxvärdet relativt snabbt stiger till nära det optimala och sedan ligger och varierar där. Anledningen till variationerna är att vi inte garanterar den bästa individen i populationen överlevnad. Populationens genomsnittsvärde stiger lite långsammare uppåt för att sedan plana ut på

en något lägre nivå. Detta utseende är typiskt för alla körningar med genetiska algoritmer.



Figur 1.6 Populationens max och genomsnittsvärde över tiden

Vi har också provat genetiska algoritmer på betydligt större problem än dessa enkla problem. Vi har kört ett stort antal körningar på olika optimeringsproblem med en kromosomlängd på upp till 6720 bitar. Vid så här pass stora problem börjar tiden bli en kritisk faktor. En körning med 300 individer och 7000 generationer tar fyra-fem timmar på en Macintosh IIfx om man har en extremt snabb avkastningsfunktion, med en ökad komplexitet hos avkastningsfunktion ökar beräkningstiden snabbt till det mångdubbla. Vår erfarenhet av dessa körningar är att det är mycket viktigt med en korrekt inställning av de olika inparametrarna som överkorsningssannolikhet, mutationssannolikhet o s v. Det är dock inte så att algoritmen misslyckas helt om man har ställt in parametrarna felaktigt, men man kan vinna många timmar på att en korrekt inställning ger en lösning som konvergerar betydligt snabbare och kommer närmre optimum.

1.4 SAMMANFATTNING

Genetiska algoritmer är en mycket robust optimeringsmetod, har man bara kodat problemet tillfredsställande hittar algoritmen alltid mycket bra värden. Genetiska algoritmer är också av en förlåtande natur, man kan göra stora avvikelser från de bästa kodningarna utan att algoritmen misslyckas, konvergensen blir dock lidande. Vid mycket stora problem är det dock viktigt att man ställer in inparametrarna rätt för att få ner beräkningstiden till en rimlig nivå. Letandet efter det optimala värdet styrs helt av en godhetsfunktion som får vara diskontinuerlig, sakna känd derivata och ha ett flertal lokala max. Till nackdelarna hör givetvis att man aldrig med säkerhet får tag i det optimala värdet och man inte heller vet hur stor avvikelsen är från detta värde. Detta behöver dock inte vara något stort problem. Vi kan göra en jämförelse med hur vi värderar en mänsklig beslutsfattare, t ex en affärsman. Hur bra han/hon anses vara beror på hans/hennes yttre konkurrens, d v s jämförelser med andra individer i samma population, vi skulle inte betrakta någons beteende som förkastligt bara för att det inte var optimalt. Slutsatsen blir alltså att det viktigaste vid studier av komplexa system inte är att hitta optimum utan istället att ständigt sträva efter förbättringar.

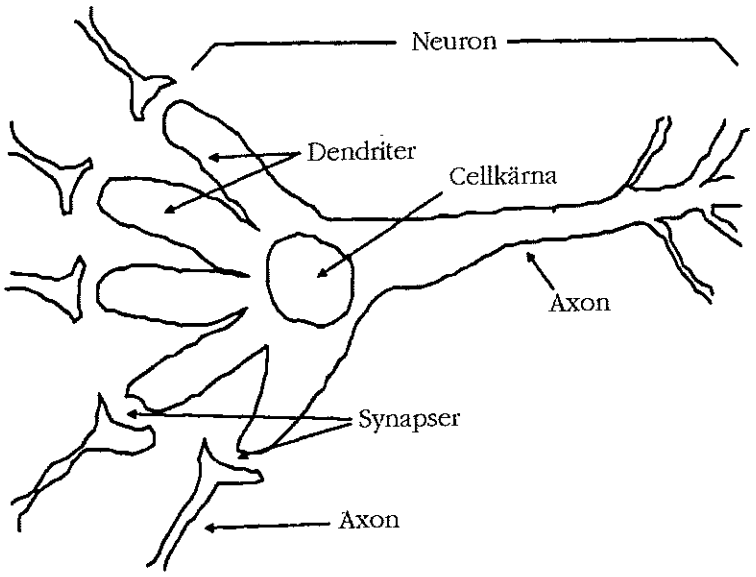
2. NEURALA NÄTVERK

I detta kapitel förklarar vi vad ett neuralt nätverk är och hur det är tänkt att användas i vår GA-optimering för att modellera kraftaffärer.

2.1 INLEDNING

Neurala nätverk är en nygammal gren av artificiell intelligens. I den artificiella intelligensens barndom när man trodde att det skulle vara möjligt att utveckla datorer som "tänkte" själva, var neurala nätverk ett populärt forskningsområde. Då Minsky och Papert 1969 publicerade boken "Perceptrons" svalnade intresset för neurala nätverk ordentligt, då de hade kommit fram till att ett nätverk med ett enkelt lager inte kunde användas till några mer avancerade uppgifter. Först 1986 då Rumelhart och McClelland publicerade "Parallel Distributed Processing" som innehöll en beskrivning av flerlagersperceptronen, tog intresset fart igen.

Neurala nätverk är ett artificiellt sätt att försöka härma hjärnans funktion. Neuronen (nervcellen) är den grundläggande byggstenen i nervsystemet och speciellt i hjärnan, se figur 2.1. Varje neuron för sig är en liten mikroprocessor som mottar och kombinerar signaler från många andra neuroner genom ingångar som kallas för dendriter. Om de kombinerade insignalerna är starka, blir neuronen elektriskt laddad och det skapas en utsignal som skickas vidare genom en celldel som kallas axon. Överföringen mellan två neuroner är kemisk men den genererar vissa elektriska sidoeffekter som man kan mäta.



Figur 2.1 Byggstenarna i nervsystemet [5].

Hjärnan består av 10^{11} neuroner, där varje neuron har kontakt med i genomsnitt 1000 andra neuroner. Axonen, utgången i en neuron, delas upp och är kopplad till andra neuroners dendriter genom kontaktytor som kallas synapser. Sändningen över en sådan kontaktyta är kemisk och mängden av signalsubstans som överförs beror dels på mängden av kemikalier som lösgör sig från axonen och dels på hur stor del av inkommande signalsubstans som dendriten upptar. Synapsens effektivitet eller styrka är det som modifieras när hjärnan lär sig. Synapsen kombinerat med bearbetningen i neuronen utgör hjärnans grundläggande minnesmekanism.

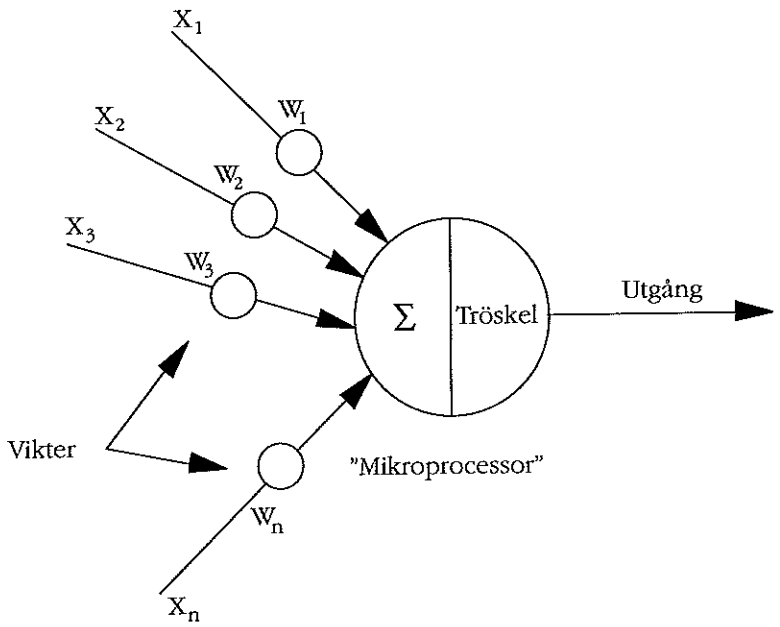
---

[5]NeuralWare, Inc: Neural Computing, 1991, s4

2.2 IMPLEMENTERING AV NEURALA NÄTVERK PÅ DATOR

2.2.1 PERCEPTRONEN

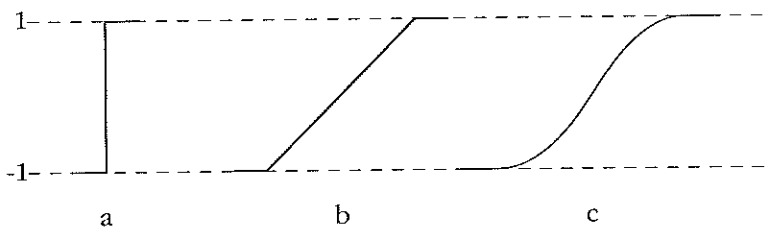
1943 föreslog McCulloch och Pitts en modell av neuronen. Denna modell gavs namnet perceptron av Frank Rosenblatt 1962, se figur 2.2. En perceptron har många ingångar vilka motsvarar neuronens dendriter. Värdet på dessa ingångar viktas ihop, oftast med en enkel summation, till en intern aktiveringsnivå för perceptronen. De sammanslagna insignalerna blir sedan modifierade av en överföringsfunktion.



Figur 2.2 Den förenklade modellen av en neuron, en så kallad perceptron

2.2.2 ÖVERFÖRINGSFUNKTIONER

Det finns flera olika överföringsfunktioner men de tre vanligaste är steg- ramp- och sigmoidfunktion, se figur 2.3. Stegfunktionen, den enklare av dem, fungerar på följande vis. När de sammanslagna insignalerna överstiger en viss nivå ger perceptronen en hög signal ut, annars ligger utgången låg.



Figur 2.3 De tre vanligaste överföringsfunktionerna Från vänster till höger a. stegfunktion b. linjärfunktion och c. sigmoidfunktion

Den linjära överföringsfunktionen ger en utsignal som är linjär mellan två nivåer. Blir aktiveringsnivån högre än den översta så blir utsignalen konstant hög och blir den lägre

än den understa så blir den konstant låg. Den sista av de tre, sigmoidfunktionen, skiljer sig ifrån de andra genom att den är kontinuerligt deriverbar. Dess överföringsfunktion är $f(x)=(1+e^{-\beta x})^{-1}$ där ß bestämmer lutningen på kurvan. Är ß oändligt stor så blir sigmoiden en stegfunktion, medan om ß är liten blir funktionen mjukare. För alla tre överföringsfuntionerna gäller att låg signal brukar vara noll eller minus ett och hög signal brukar vara ett. I vårt fall valde vi att representera låg signal med minus ett.

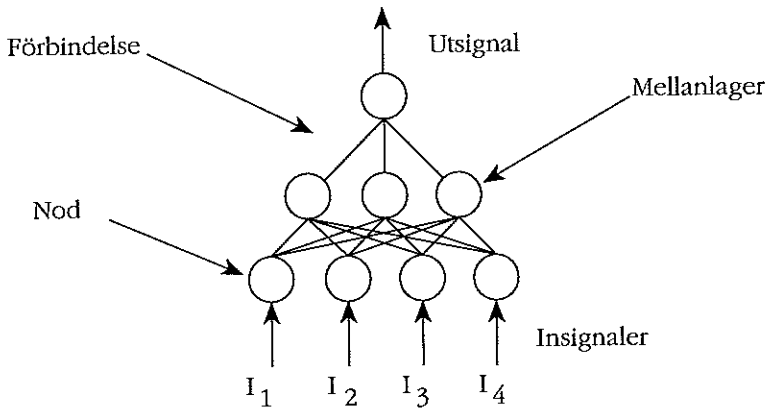
2.2.3 PERCEPTRON NÄTVERK

Utgången från en perceptron kan vara kopplad till ingångarna på andra perceptroner. De olika ingångarna på en perceptron har olika vikter. Dessa vikter skall symbolisera styrkan hos synapserna. Eftersom varje ingång har en tillhörande vikt blir ingångsvärdena modifierade innan de summeras ihop inne i perceptronen. Trots att den summerande funktionen är en viktad summation är inte den förenklade funktionen av en neuron speciellt användbar i sig själv utan det är först när man kopplar ihop flera perceptroner till ett nätverk som det börjar bli intressant.

Ett neuralt nätverk består av många perceptroner som sätts samman efter den princip vi nämnt ovan. Perceptroner är oftast organiserade i grupper som kallas för lager. Ett typiskt nätverk består av ett antal lager som är fullständigt eller slumpmässigt anslutna med varandra. Det finns oftast två lager med förbindelser med yttervärlden. Ett ingångslager där data presenteras för nätverket och en utgång som ger utsignalen eller utsignalerna för den givna insignalen. Lager som ligger mellan ingången och utgången kallas för dolda lager.

2.2.4 BACKPROPAGATIONNÄTVERK

Backpropagation är en övervakad inlärningsalgoritm, som använder sig av ett nätverk som består av två eller flera perceptronlager. De olika perceptronlagrerna är antingen fullständigt eller delvis förbundna med varandra, se figur 2.4.

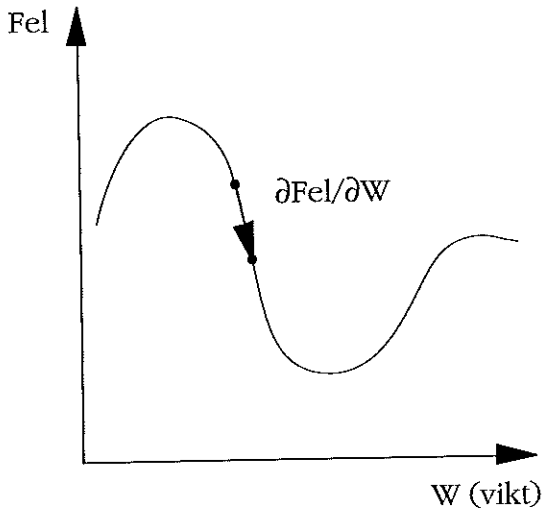


Figur 2.4 Figuren visar ett fullständigt förbundet trelagers backpropagationnätverk

Algoritmen i ett backpropagationnätverk använder sig av "steepest descent"-metoden för att finna vikter som minimerar felen på nätverkets utsignaler. Man deriverar bara det fel man får med avseende på vikterna i nätverket ($\partial fel/\partial W$) och ändrar sedan vikterna åt det håll derivatan pekar, se figur 2.5.

Karakteristiskt för ett backpropagationnätverk är att perceptronernas överförings-funktioner oftast är kontinuerliga och olinjära(sigmoidfunktion). Vid inlärning presenterar man för nätverket ett indata/utdatapar och backpropagationalgoritmen uppdaterar därefter perceptronernas vikter så att nätverket skall kunna reproducera en

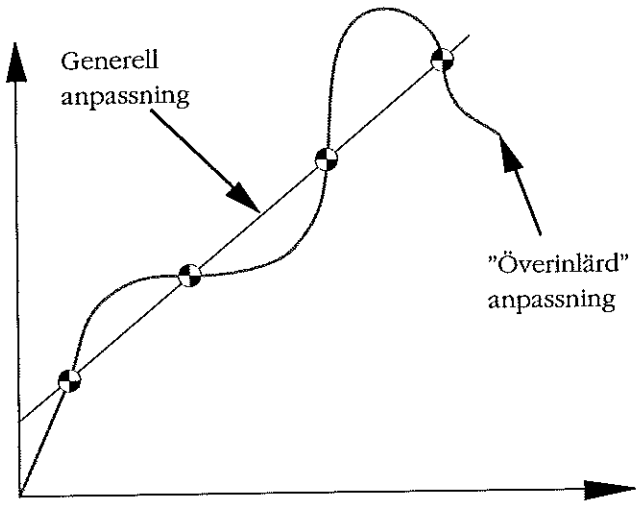
utsignal som svarar mot motsvarande insignal. Detta förfarande upprepas sedan för alla indata/utdatapar tills det att nätverket "känner igen" alla indatapar.



Figur 2.5 Bilden visar felet som en viss neuron gör för olika vikter på sina insignaler. W utgör egentligen en vektor med samma dimension som antalet insignaler till neuronen. Bilden visar även hur felet kan minskas om man ändrar på vikterna.

2.2.5 ÖVERINLÄRNING

En viktig egenskap hos neurala nätverk är dess förmåga att generalisera indata det tidigare inte har stött på. Tidigare försök har visat sig att backpropagationnätverk är mycket bra på att generalisera om man undviker att gå i en och annan fallgrop. En av fallgroparna är överinlärning.



Figur 2.6 Figuren visar hur ett överinlärt nätverk har memorerat alla insignal-, utsignalkombinationer och hur man har misslyckats med att få den generella anpassningen som man vill få fram med hjälp av det neurala nätverket.

Överinlärning uppstår då man har tränat ett backpropagationnätverk under en för lång tid. Nätverket uppnår då en nivå där det blir sämre på att känna igen en testmängd som

det inte har använt under upplärningsfasen. Detta beror på att nätverket har börjat att memorera de olika insignal/utsignalparen. Överinlärning kan jämföras med en kurvanpassning som anpassas till exakt givna punkter se figur 2.6. Det finns några olika metoder att komma runt detta problem.

En metod är att helt enkelt sluta lära upp nätverket när man märker att nätverket blir sämre på att känna igen testmängden. En annan metod går ut på att addera brus till insignalen och en tredje metod är att minska antalet perceptroner och gömda perceptronlager. Trots risken att nätverket kan komma att memorera de olika insignalsalternativen är backpropagationnätverken förmodligen den nätverkstyp som idag är bäst på att generalisera.

2.3 KRAFTAFFÄRER

Efter denna inledning till neurala nätverk skall vi förklara vad syftet med dessa skall vara i detta examensarbete. Då det saknas en enkel matematisk modell för affärerna med Norge och Danmark skulle vi försöka bygga en modell med hjälp av neurala nätverk. Dels för att det är en ganska enkel metod att använda, och dels för att det har gjorts ett examensarbete om "Belastningsprognosering i elnät med hjälp av neurala nätverk"[6] som visade sig prediktera framtida värden bättre än de tidserieanalyser som gjorts på detta problem. Efter en del överväganden drog vi slutsatserna att ett liknande nätverk kanske kunde vara lämpligt att använda för att prediktera de utbytespriser vi behöver för att beräkna vår godhetsfunktion.

Sedan april 1992 har det skett en del förändringar när det gäller affärerna med Norge. Tidigare har man utgått ifrån mittprisprincipen. Köparen angav sin marginalkostnad för att producera kraften själv och säljaren sin marginalkostnad för att producera den extra kraften, därefter satte man priset mitt emellan dessa båda priser och delade på så sätt på vinsten.

Numera så sker handeln på en så kallad kraftbörs där man köper och säljer kraft till marknadspriser och där den som betalar bäst får köpa kraften. Detta tror man kommer att leda till att tillgången på information om de olika företagens produktionsmix blir sämre, samt att öppenheten kraftföretagen emellan kommer att minska. Omläggningen av krafthandeln resulterade i att vi fick relativt lite data att träna våra nätverk med. Trots detta valde vi att försöka modellera kraftbörsen.

Då där finns flera utbytespunkter mellan Sverige och Norge som Sydkraft använder valde vi ut den som hade haft mest utbyte under de veckorna som finns tillgängliga för att få så mycket data som möjligt. Vi fick data från veckorna 14-34 och valde bort tre veckor från dessa för att ha något att testa vårt nätverk på.

2.4 IMPLEMENTERINGEN AV KRAFTAFFÄRER PÅ DATOR

När man vill ha med tid som indata till ett nätverk så uppstår det ett problem. Man kan nämligen inte representera till exempel timmar med talen 1-24 p g a att nätverket skulle uppfatta timme 24 som mer värd än timme 1. Då nu varje timme är unik och måste särbehandlas av nätet, så låter vi varje timme representeras av en nod i nätet. Klockan 1 anges då med en etta i den första noden och med nollor i de 23 övriga. Man kan på samma sätt bygga upp vecko- och dagnummer.

Att representera veckonummer med denna metod skulle kräva 52 innoder vilket är lite väl mycket med tanke på att vi är intresserade av att få ned beräkningstiden till ett minimum i det färdigtränade nätverket. Ett bättre sätt att representera veckonumren är

$$\sin(2\pi * \text{vecka}/52) \text{ och } \cos(2\pi * \text{vecka}/52)$$

---

[6] Ericsson Olof, Haglund Ola, Belastningsprognosering i elnät med hjälp av neurala nätverk, LTH, Lund, LU-CS-EX:91-24

som insignaler till två noder. Vi får då en cirkulär representation där veckonumret representeras med en punkt som vandrar runt enhetscirkeln.

De indata vi matar in är följande:

- Veckonummer, två stycken noder med sinus och cosinus på veckonumret.
- Dagnummer, sju stycken noder med ett i aktuell dag och nollor i övriga.
- Timnummer, 24 stycken noder med ett i aktuell timme och nollor i övriga.
- Kvantiteten elkraft som vi köper för tillfället.
- Den last som Sydkraft har för tillfället.
- Aktuella vattenmagasinslägen i Norge i denna veckan.
- Förändringar i vattenmagasinen sedan föregående vecka.

Som utdata får vi priset per MWh el som vi köper. Vi fick alltså sammanlagt 37 innoder.

Det finns ett antal olika tumregler för hur många noder man sammanlagt ska ha i mellanlagren. Antalet noder eller vikter är ett mått på hur mycket information som kan lagras i nätet. Väljer man för få noder kan inte all önskad information lagras och nätet kommer inte att generalisera speciellt bra. Skulle man istället välja för många så kan man få problem med överinlärning.

En tumregel säger att man inte ska använda fler anslutningar i nätet än man har olika insignalsset. Om man har k insignalsnoder och ett mellanlager med j noder och en utsignalnod samt ett fullt anslutet nät, så är det j*(k+1) anslutningar i nätet. En annan tumregel är att man ska ha 50-70% av totala antalet insignal- och utsignalsnoder i mellanlagret, en tredje regel säger det minsta av de två tumreglerna.

2.5 Resultat

Vi experimenterade med olika mellanlagerstorlekar och körde ett antal olika nätverk. Efter en del försök upptäckte vi att nätverken inte generaliserade speciellt bra, så vi plockade bort alla de rader i kolumnen med elpriset som innehöll nollor. Detta gjorde vi för att det visade sig att där finns ett antal timmar på en månad som Sydkraft inte hade något elpris för Norgekraften och vi tror att det kan påverka nätets inlärnings-förmåga på ett negativt sätt. Efter detta ville nätverken lära sig lite bättre men ingen av våra nätverk fick ner RMS-felet (minsta-kvadrat-felet) till mindre än till 25%, vilket är på tok för högt. Ett godtagbart RMS-fel ligger på högst 5%.

Det som var mest intressant för vårt optimeringsproblem var att se hur lång tid det tar att anropa den C-kod som NeuralWorks genererar. Vi tog fram kod både med matte-processor och utan för ett nätverk med konfigurationen 37-10-1 och mätte tiden. Det visade sig att ett anrop på en Quadra 700 d v s en 25 Mhz 68040 tog ca 7,3 ms utan och 3 ms med matteprocessor. Detta ska jämföras med anropet för vår kärnkraftsfunktion som tar cirka en mikrosekund. Av detta drar slutsatsen att man nog ska tänka sig för och inte anropa denna procedur onödigt många gånger med tanke på den stora mängd anrop som sker till vår godhetsfunktion. En lösning man kan använda för att reducera tiden, är att räkna ut kostnaden för de diskreta intervall som man har valt att implementera i sin algoritm och stoppa in talen i någon form av tabell.

2.6 Sammanfattning

För att överhuvudtaget kunna använda neurala nätverk i GA optimeringen måste man göra vissa förbättringar. Det första man skulle kunna pröva för att förbättra nätverkets generalisering är att stoppa in Sydkrafts egen kraftproduktion som inparameter och på så sätt hoppas att man skulle få bättre värden. Att våra testveckor var utplockade mitt i testmängden, på grund av att vi inte kunde få fram Norgepriset för dessa veckor, kan också ha försämrat våra resultat. Denna slutsats drar vi av att man inom tidserieanalys

teorin har som regel att man bör ha en sammanhängande period under modell-framtagandet och sedan använda en annan period för utprovandet av modellen. Nätverket borde bli bättre om man hade försökt lära upp nätverket med en större mängd data.

För att få ned beräkningstiden för nätverket borde man försöka minska ner antalet noder i nätverket och på så sätt minska beräkningstiderna för godhetsfunktionen. Man kan ganska enkelt lista ut att för varje extra nod i mellan lagret så ökar antalet vikter man ska räkna ut med n*(1+antalet noder i insignalslagret). I vårt problem skulle vi till exempel kunna representera även veckodagarna och timmarna med sinus och cosinus värden som vi gjorde med veckonummrena. På så sätt hade vi minskat ner antalet insignalsnoder till tio istället för våra ursprungliga 37. Om man då skulle ta med Sydkrafts egen kraftproduktion skulle man få ett nätverk med elva insignalsnoder. Detta i sin tur hade snabbat upp anropen för godhetsfunktionen betydligt.

Man kunde även implementera godhetsfunktionen i någon form av tabell för de diskreta värden man väljer för sin kromosomimplementering. För att få en så noggrann beräkning av priserna skulle man kunna införa en tabell för respektive dag eller olika för hög respektive lågbelastning och på så sätt ändå inte öka tidsåtgången. Ett problem uppstår om man ska ta fram tabellerna i förväg. Om man ska ta med Sydkrafts egen produktion som inparameter så måste man skatta denna parameter eftersom det är tänkt att GA ska ta fram denna parameter.

Slutsatsen är i detta fall där tidsaspekten är speciellt viktigt bör man testa att sig fram tills man finner ett nätverk som generaliserar bra men som har så få insignalsnoder som möjligt.

3. KODNING OCH INSTÄLLNINGAR AV GA

I detta kapitel diskuterar vi olika inparametrar, deras genomslag på konvergensen och lämpliga val. Vi diskuterar även olika kodningar av kromosomen och problemet, samt slutligen algoritmkomplexiteten.

3.1 INPARAMETRAR

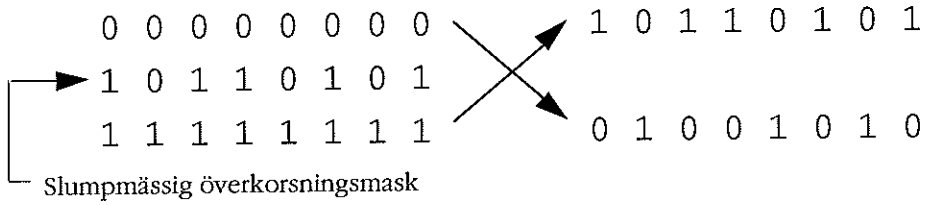
För att förbättra GA:s prestanda är det mycket viktigt att man ställer in algoritmens olika parametrar på bästa möjliga sätt. Inställningarna är ofta mycket problemberoende men vissa generella regler anses gälla. Nedan följer avsnitt för de viktigaste inparametrarna i vårt utökade GA-program.

3.1.1 ÖVERKORSNINGSSANNOLIKHET

Överkorsningssannolikheten är den sannolikhet med vilken två utvalda föräldrar verkligen parar sig med varandra. Om de inte parar sig med varandra flyttas de över till nästa generation i oförvanskat skick. Detta innebär att en överkorsningssannolikhet lika med noll inte utvecklar individerna överhuvudtaget. De Jong visade med hjälp av empiriska undersökningar i sin avhandling[7] att överkorsningssannolikheten bör ligga högt, mellan 0.5 och 1. Vi har själva med framgång använt oss av överkorsnings-sannolikheter kring 0.8-0.9.

3.1.2 ÖVERKORSNINGSFUNKTIONER

Vi har implementerat ett flertal olika överkorsningsfunktioner; enkel, multipel, uniform, analog och kombinationer av dessa. Enkel överkorsning är den överkorsningsfunktion som beskrivs i kapitel 1. Vid multipel överkorsning går man ett steg längre och använder sig av flera slumpmässigt valda överkorsningspositioner istället för endast en, i övrigt fungerar de på samma sätt. Man kan utveckla detta ytterligare till den uniforma överkorsningen där även antalet överkorsningspositioner är slumpmässigt valda. Uniform överkorsning går till så att man skapar sig en slumpmässig överkorsningsmask, därefter skapas barn1 genom att man kopierar förälder2:s gener för varje position där masken är ett, där masken är noll kopieras förälder1:s gener. Barn2 skapas sedan på motsvarande sätt genom att man inverterar masken. Figur 3.1 nedan förklarar tillvägagångssättet.



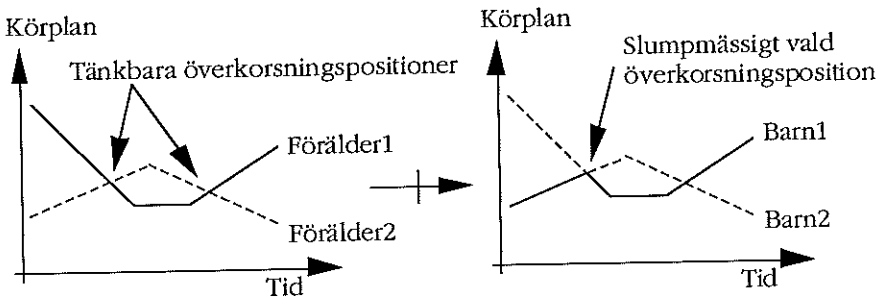
Figur 3.1 Den uniforma överkorsningsoperatorn.

Den analoga överkorsningsfunktionen skiljer sig från de övriga genom man styr valet av överkorsningsposition genom någon överordnad regel. Vanligtvis styr man valet av överkorsningsposition så att man undviker kraftiga hopp i körplanen. Man jämför de båda föräldrarnas körplaner med varandra, i positioner där de har samma värden kan man tillåta överkorsning. Någon av de tänkbara överkorsningspositionerna väljs slumpmässigt ut. Därefter sker överkorsningen på vanligt sätt. Figur 3.2 nedan beskriver den analoga överkorsningsfunktionen. Syftet med denna överkorsningsfunktion är att

---

[7]De Jong Kenneth A, An Analysis of a Class of Genetic Adaptive Systems, George Mason University, 1975

reducera brus och undvika kraftiga hopp i körplanerna. Tyvärr är vår erfarenhet att den även minskar konvergenshastigheten.

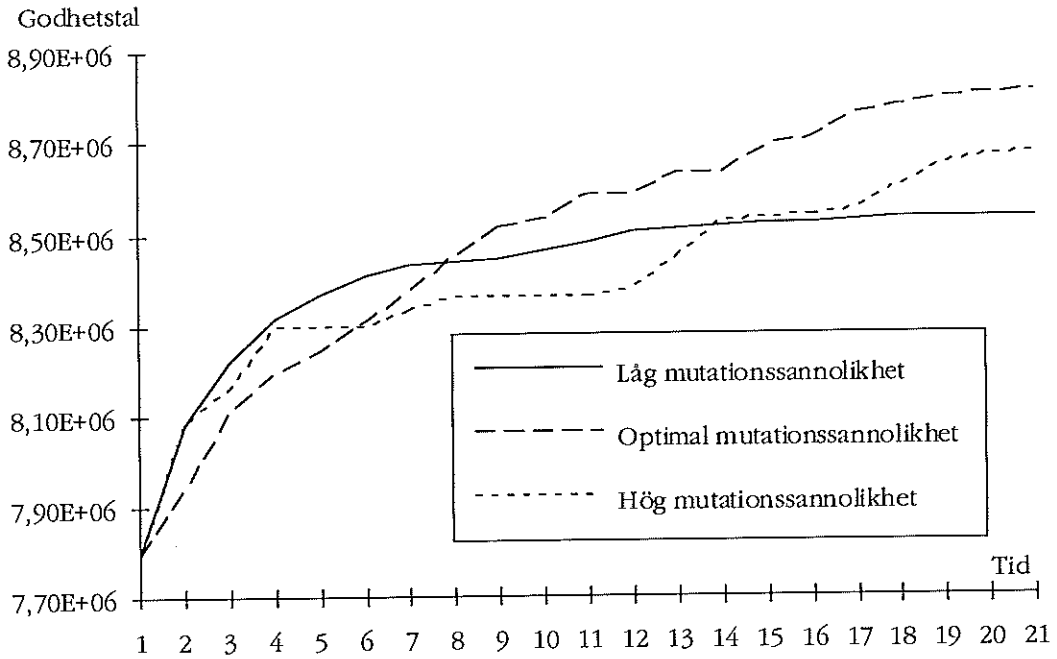


Figur 3.2 Den analoga överkorsningsoperatorn.

Dessutom har vi använt oss av olika kombinationer av ovanstående överkorsningsfunktioner på olika delar av kromosomen. Våra resultat tyder på att den uniforma överkorsningen ger bäst resultat. Detta överensstämmer med Gilbert Syswerdas[8] empiriska resultat som visar att den uniforma överkorsningen är överlägsen enkel överkorsning på i stort sett alla problemtyper.

3.1.3 MUTATIONSSANNOLIKHET

Vi har vid våra undersökningar empiriskt kommit fram till att mutationssannolikheten är en av de viktigaste parametrarna att ställa in rätt för att uppnå en snabb konvergens. Mutationssannolikheten är sannolikheten för att en viss gen skall muteras under en generation. Denna sannolikhet skall väljas till ett mycket lågt tal, man bör välja det så att det endast blir en eller ett fåtal mutationer per kromosom under en generation. Mutationssannolikheten blir därmed beroende av kromosomlängden.



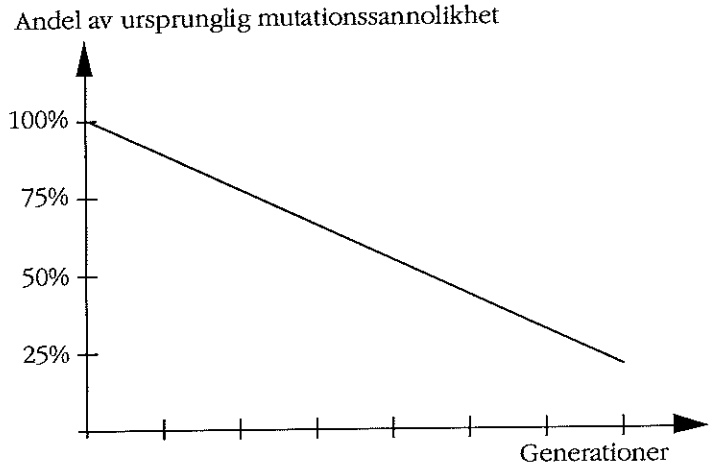
Figur 3.3 Effekterna på konvergensen för olika mutationssannolikheter.

[8]Syswerda Gilbert, Uniform Crossover in Genetic Algorithms, Proceedings of the Third International Conference of Genetic Algorithms, Morgan Kaufmann Publishers, 1989

I figur 3.3 ser man effekterna på konvergensen för olika mutationssannolikheter. Om man väljer en för låg mutationssannolikhet leder detta till en för tidig konvergens vid en lägre godhetsnivå. Väljer man istället en alltför hög mutationssannolikhet medför detta ett ryckig konvergens föranledd av lyckosamma mutationer. Båda dessa mutationssannolikheter är underlägsna den optimala som ger en jämn och stadig konvergens som närmar sig problemets optimala lösning.

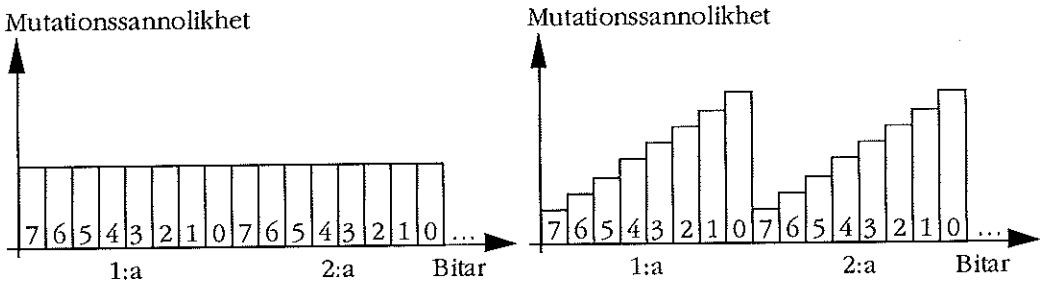
3.1.4 MUTATIONSFUNKTION

Mutatationsfunktionen är en funktion som förändrar mutationssannolikheten, antingen över tiden, mellan olika gener i kromosomen eller kombinationer av dessa. Vanligt är någon form av avtagande mutationssannolikhet över tiden. Detta beror på att i början av en körning är populationens individer relativt dåliga. En mutation förbättrar då med stor sannolikhet en viss individen. I slutet av en körning är istället individerna mycket väl anpassade till problemet och endast små mutationer förbättrar dem ytterligare.



Figur 3.4 Den linjärt avtagande mutationsfunktionen.

Man kan även ändra mutationssannolikheten inom kromosomen så att vissa gener har högre respektive lägra mutationssannolikhet. Vanligtvis ökar man mutationssannolikheten för den minst signifikanta biten och minskar den för den mest signifikanta biten. På detta sätt ökar man sannolikheten för att en mutation endast orsakar mindre förändringar. Ett annat sätt att åstadkomma en liknande effekt är att använda graykodade kromosomer. En mutation ger då alltid upphov till förändringar om endast ett binärt steg.



Figur 3.5 Den konstanta och den bitvisa mutationsfunktionen.

Vi jämförde fem olika typer av mutationsfunktioner, konstant mutationssannolikhet, linjärt avtagande, bitvis förändrad mutationssannolikhet samt de två förstnämnda med graykodade kromosomer. Vårt resultat blev att den linjärt avtagande icke-graykodade gav bäst resultat. Det visade sig även att den konstanta mutationsfunktionen var sämst. Våra resultat stöds även av en undersökning gjord av Terence C Fogarty[9] som visat att förändringar av mutationssannolikheten över tiden och/eller över heltalsrepresentationen påtagligt förbättrar GA:s konvergenshastighet.

3.1.5 MIGRATIONSSANNOLIKHET

Migrationssannolikheten är den sannolikhet med vilken en individ emigrerar från sin hempopulation till en annan slumpmässigt vald delpopulation. Denna sannolikhet skall väljas relativt lågt. En stor migrationssannolikhet resulterar i att delpopulationerna fungerar som en enda stor population och man går miste om de fördelar en uppdelad population ger. Emellertid kan en alltför liten migrationssannolikhet orsaka samma problem som en liten population, nämligen inavel och otillräcklig konvergens. Vi har experimenterat oss fram till en migrationssannolikhet runt 0.1.

3.1.6 STRAFFUNKTIONER

Vid GA optimering uppkommer naturligt lösningar som är otillåtna, d v s de överträder någon eller några av problemets begränsningar. Det kan till exempel vara att man utför alltför kraftiga upp- eller nedregleringar av kärnkraft. Dessa lösningar innehåller, trots att de är felaktiga, värdefullt genetiskt material som man ej vill förlora. Därför dödar man inte dessa individer genast utan straffar istället deras godhetsfunktion.

Det är viktigt att straffet står i proportion till överträdelsens storlek, ju större överträdelser och felkörningar, desto högre straff och därmed lägre godhetstal. Straffen skall också väljas så att de avskräcker GA från överträdelser av uppsatta begränsningar och inte skapar otillåtna lösningar med högre godhetstal än den bästa tillåtna lösningen.

Man kan även låta straffunktionen variera över tiden. Vi testade en linjärt ökande funktion och jämförde den med en funktion med konstant straffnivå. Tanken bakom den linjärt ökande straffunktionen är att ett lågt straff i början av körningen skulle öka konvergenshastigheten medan man först i slutet skulle sortera ut felaktiga körningar med hjälp av höga straff. På våra problem visade sig dock att den konstanta straffunktionen var bäst.

3.1.7 POPULATIONENS BETYDELSE

Totalpopulationens storlek och dess fördelning mellan olika delpopulationer är två mycket viktiga faktorer för en lyckad GA-körning. Det man i praktiken dessutom måste ta hänsyn till är internminnets storlek som begränsar totalpopulationens storlek uppåt. Vid mycket stora problem leder detta till att man måste använda sig av en population som är mindre än den optimala.

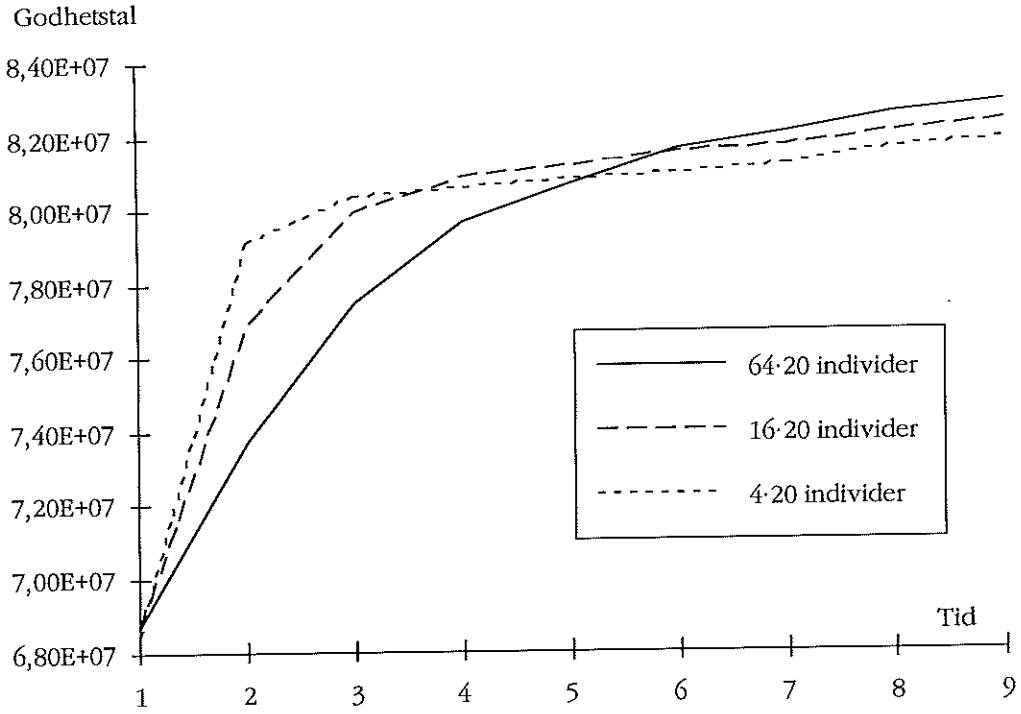
Fördelningen av totalpopulationens individer mellan delpopulationerna har också ett optimum som beror på problemets karaktär. I de problem vi presenterar i kapitel fyra och fem har vi med framgång använt oss av delpopulationsstorlekar kring 20-30 individer. Därutöver har vi inte undersökt denna parameter.

Figur 3.6 nedan visar konvergensen för tre olika populationsstorlekar. Det bör noteras att problemet vi gjorde dessa körningar på var mycket stort och bilden därför kan förleda en att tro att ju större population desto närmre kommer man optimum. På mindre problem finner man dock en optimal populationstorlek som leder till den snabbaste

---

[9]Fogarty Terence C, Varying the Probability of Mutation in Genetic Algorithm, Proceedings of the Third International Conference of Genetic Algorithms, Morgan Kaufmann Publishers, 1989

konvergensen, större populationer leder inte till en högre godhetsnivå utan tar bara längre tid.



Figur 3.6 Totalpopulationens betydelse för konvergensen

Anledningen till att kurvorna ser ut som ovan är att i en liten population finns normalt inte allt det genetiska material som behövs för att konvergera mot optimum, resultatet blir en konvergens mot en alltför låg godhetsnivå. I en stor population sprider sig exempelvis en lyckad mutation långsamt i populationen. Därför blir konvergensen långsammare.

3.2 KROMOSOMKODNING

GA skiljer sig från vanliga optimeringsmetoder genom att man använder sig av en kod av parametrarna och inte med parametrarna själv. Valet av denna kod har mycket stort genomslag på GA:s uppförande. Därför bör man lägga ner stor möda vid utformningen av koden för att försäkra sig om en god konvergens.

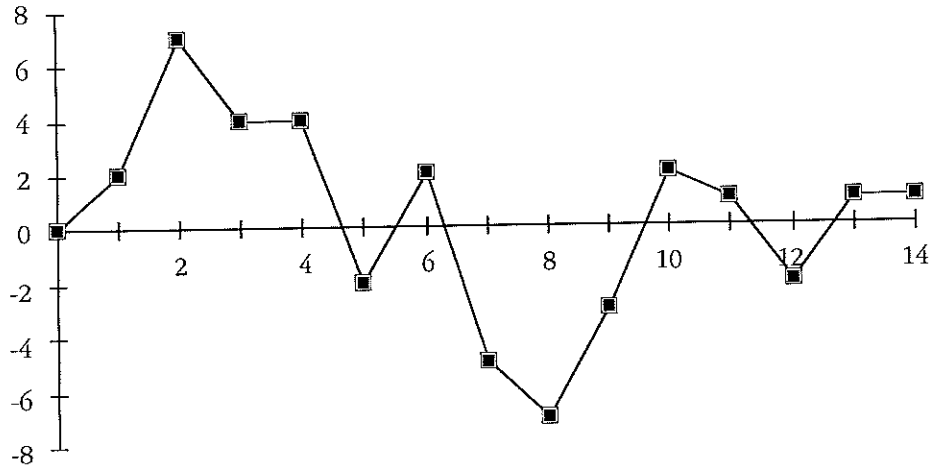
3.2.1 BEROENDE INOM KROMOSOMEN

Ett allvarligt problem som kan uppkomma vid kodningen är att det uppstår beroende mellan kromosomens olika gener. Ofta är detta oundvikligt men man bör känna problemet så man inte blir överraskad av den långsamma konvergensen. För att exemplifiera problemet har vi konstruerat följande problem. Vi har skapat nedanstående 14-punkters kurva, se figur 3.7, och vill att de genetiska algoritmerna avbildar denna. Vi har valt godhetsfunktion till:

$$10000 - \sum_{i=1}^{14} (\text{avvikelsen från den rätta kurvan i punkten[i]})^2$$

Anledningen till att vi subtraherar summan av de kvadratiska avvikelserna med 10000 är för att transformera om problemet från ett minimeringsproblem till ett maximerings-

problem som genetiska algoritmer kräver. Talet 10000 är valt så att godhetsfunktionen alltid är ett positivt tal. Om de genetiska algoritmerna lyckas avbilda kurvan perfekt når godhetsfunktionen sitt optimum på 10000.



Figur 3.7 Kurvan vi försöker avbilda.

Vi har valt två olika kromosomrepresentationer av kurvan, en med ett inbyggt beroende och en utan. Den senare består av 14 stycken 4-bitars binärkodade tal som var och en representerar talen -8 till +7. Varje tal kopplas till en specifik punkt på kurvan, d v s de första 4 bitarna kopplas till punkt 1 på kurvan, nästa 4 bitar till punkt 2, o s v.

Den förstnämnda kromosomrepresentationen består också av 14 stycken 4-bitars binärkodade tal som var och en representerar talen -8 till +7. För att skapa ett beroende genom kromosomen låter vi dock inte talen direkt kopplas till kurvan utan istället betyda förändringar från föregående position på kurvan. Med andra ord, de första 4-bitarna ger förändringen från origo och relaterar denna till punkt 1 på kurvan, nästa 4 bitar ger förändringen från denna position och relaterar denna till punkt 2 på kurvan, o s v. Detta medför att varje position blir beroende av de föregående positionerna.

En körning med kromosomrepresentationen utan beroende ser ut som följer. Vi ser att man nått det optimala värdet och därmed skapat en perfekt avbildning av kurvan på mindre än 80 generationer .

```
GA-parameters
-------------------------------------------------------------------------
Crossover probability:              0.80000
Mutation probability:               0.00700
Migration probability:              0.10000
Maximum number of generations:      80
Number of subpopulations:           4
Size of each subpopulation:         15
Chromosomelength (bits):            56
-------------------------------------------------------------------------
Generation:     1,   Max: 9.8410e+03,   Min: 9.3180e+03,   Average: 9.6107e+03
Generation:    20,   Max: 9.9830e+03,   Min: 9.8660e+03,   Average: 9.9525e+03
Generation:    40,   Max: 9.9940e+03,   Min: 9.9350e+03,   Average: 9.9829e+03
Generation:    60,   Max: 9.9980e+03,   Min: 9.7380e+03,   Average: 9.9827e+03
Generation:    80,   Max: 1.0000e+04,   Min: 9.8010e+03,   Average: 9.9797e+03
-------------------------------------------------------------------------
Number of mutations: 3526,  crossover: 1780,  migrations: 434
Current time: 16:22:56 - 1992.08.12,  Executiontime: 6 sek
```

En körning med kromosomberoende ger inte ett lika lysande resultat. Efter 400 generationer har fortfarande inte optimum hittats. Vi ser också att algoritmen "fastnar" på samma max-värde under lång tid.

```
GA-parameters
-------------------------------------------------------------------------------
Crossover probability:              0.80000
Mutation probability:               0.00700
Migration probability:              0.10000
Maximum number of generations:      400
Number of subpopulations:           4
Size of each subpopulation:         15
Chromosomelength (bits):            56
-------------------------------------------------------------------------------
Generation:    1,  Max: 9.8390e+03,  Min: 4.0560e+03,  Average: 8.4563e+03
Generation:   20,  Max: 9.9220e+03,  Min: 9.2230e+03,  Average: 9.8269e+03
Generation:   40,  Max: 9.9530e+03,  Min: 7.9510e+03,  Average: 9.8249e+03
Generation:   60,  Max: 9.9690e+03,  Min: 9.4740e+03,  Average: 9.9064e+03
Generation:   80,  Max: 9.9770e+03,  Min: 8.4550e+03,  Average: 9.8480e+03
Generation:  100,  Max: 9.9810e+03,  Min: 9.6080e+03,  Average: 9.9323e+03
Generation:  120,  Max: 9.9810e+03,  Min: 9.0250e+03,  Average: 9.9218e+03
Generation:  140,  Max: 9.9810e+03,  Min: 8.8430e+03,  Average: 9.9098e+03
Generation:  160,  Max: 9.9810e+03,  Min: 9.6130e+03,  Average: 9.9312e+03
Generation:  180,  Max: 9.9810e+03,  Min: 9.1160e+03,  Average: 9.9086e+03
Generation:  200,  Max: 9.9810e+03,  Min: 9.6890e+03,  Average: 9.9487e+03
Generation:  220,  Max: 9.9870e+03,  Min: 8.9240e+03,  Average: 9.8947e+03
Generation:  240,  Max: 9.9870e+03,  Min: 9.2950e+03,  Average: 9.9292e+03
Generation:  260,  Max: 9.9870e+03,  Min: 9.0070e+03,  Average: 9.8840e+03
Generation:  280,  Max: 9.9870e+03,  Min: 8.9370e+03,  Average: 9.8907e+03
Generation:  300,  Max: 9.9870e+03,  Min: 9.1870e+03,  Average: 9.9317e+03
Generation:  320,  Max: 9.9870e+03,  Min: 9.1170e+03,  Average: 9.9260e+03
Generation:  340,  Max: 9.9870e+03,  Min: 9.3450e+03,  Average: 9.9394e+03
Generation:  360,  Max: 9.9870e+03,  Min: 9.0150e+03,  Average: 9.8995e+03
Generation:  380,  Max: 9.9870e+03,  Min: 9.1340e+03,  Average: 9.8960e+03
Generation:  400,  Max: 9.9870e+03,  Min: 9.3250e+03,  Average: 9.9694e+03
-------------------------------------------------------------------------------
Number of mutations: 17364,  crossover: 8934,  migrations: 2176
Current time: 16:24:47 - 1992.08.12,  Executiontime: 14 sek
```

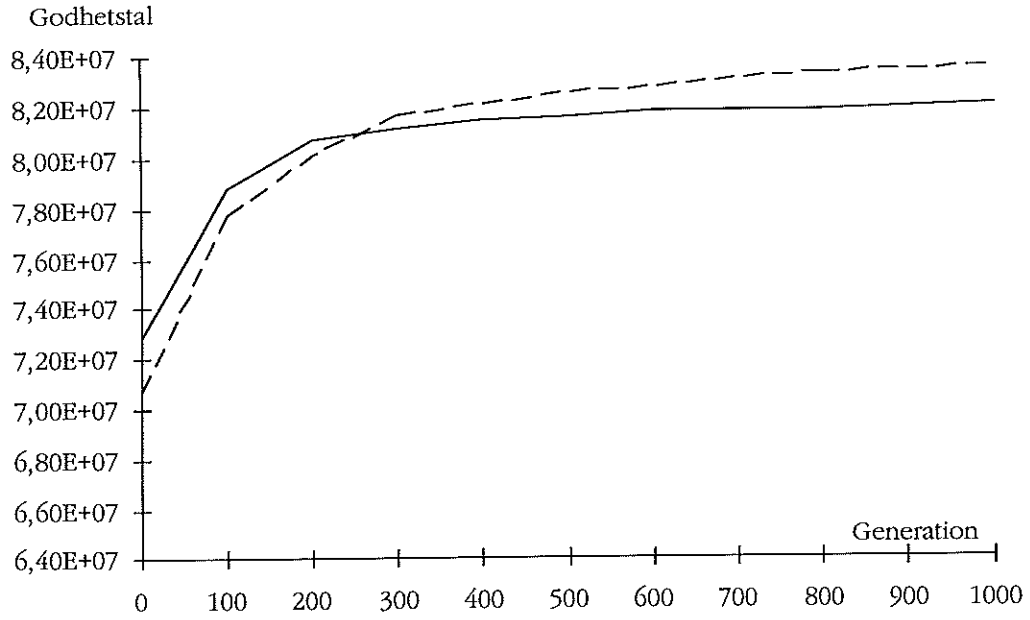
Detta exempel visar på de problem man ställs inför då man skapar beroende mellan kromosomens olika gener. Exemplet ovan har bara ett enkelt beroende bakåt i kromosomen. Svårigheterna för GA ökar naturligtvis om man har beroenden som löper på båda hållen i kromosomen eller om man har andra typer av korsvisa beroenden inom kromosomen.

3.2.2 UPPLÖSNING OCH KROMOSOMLÄNGD

Kromosomlängden beror helt på kodningen av problemet och problemets storlek. I de problem vi presenterar i kapitel fyra och fem beror kromosomlängden på antalet kraftverk och övriga objekt, antalet timmar i optimeringsperioden och på upplösningen. Med upplösning menar vi med hur många bitar en körnivå i ett kraftverk el dyl representeras med. Ett extremfall är 1-bits upplösning som ger on/off-reglering. Vi har använt oss av 4-bitars respektive 8-bitars upplösning.

En mycket intressant iakttagelse vi gjort är att en ökning av upplösningen ökade konvergenshastigheten markant. Detta strider mot vad man skulle kunna förvänta sig vid en första anblick. Varje extra bit i kromosomen fördubblar antalet möjliga lösningar, d v s GA:s sökområde fördubblas för varje extra bit. Trots detta konvergerar problemet betydligt snabbare. Det kostar dock, tiden per generation ökar i vårt problem med cirka 15 procent när man fördubblar kromosomlängden. Men, på grund av den ökade konvergenshastigheten minskar man ändå tidsåtgången. Att tiden inte ökar mer per generation beror på att det är godhetsfunktionen som tar största delen av beräkningstiden per generation, och tiden för denna beror ej på upplösningen utan

bara på antalet objekt och timmar i problemet. Godhetstalet som funktion av antalet generationer visas nedan i figur 3.8 för två körningar med 4-bitars respektive 8-bitars upplösning.



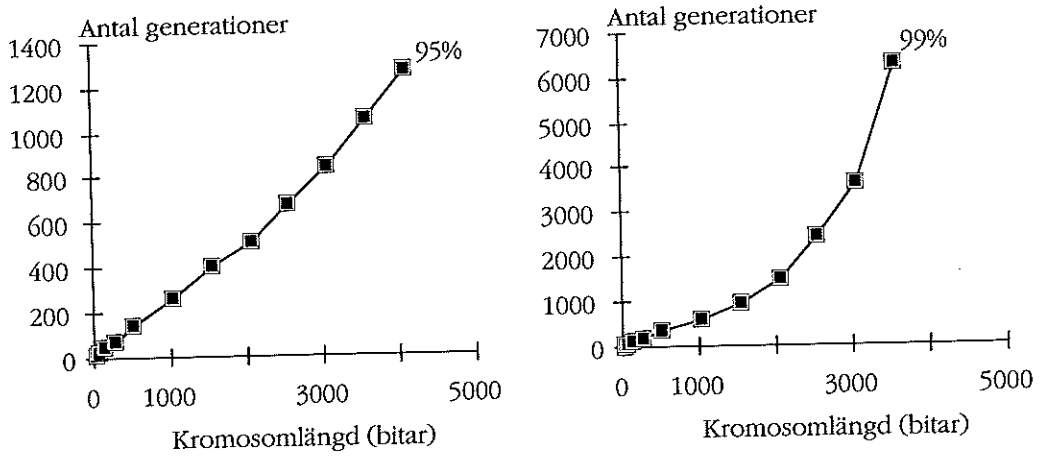
Figur 3.8 Konvergens för fyra-bitars (hel) och åtta-bitars (streckad) upplösning.

3.3 ALGORITMKOMPLEXITET

Med en algoritms komplexitet menas här tidsåtgång och minnesåtgång som funktion av problemets storlek. För minnesåtgången gäller att problemets storlek bestäms av väsentligen två faktorer, nämligen kromosomlängden(n) och antalet individer i populationen. Om man håller populationsstorleken fix blir den asymptotiska minnesåtgången M(n) = O(n).

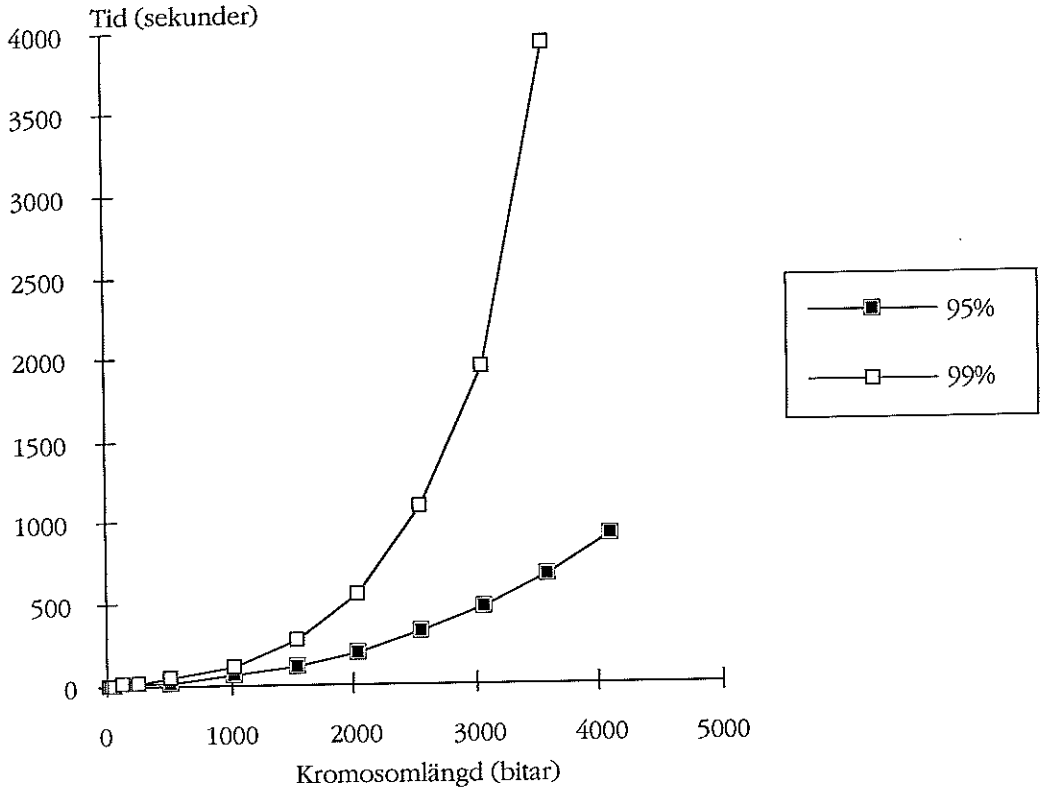
Det är betydligt svårare att bestämma den asymptotiska tidsåtgången för algoritmen eftersom tidsåtgången beror på ett par samverkande faktorer. Ökar man kromosomlängden ökar tiden för en ny generations skapande linjärt med kromosomlängden, en ökning av kromosomlängden kräver dock även ett större antal generationer för att konvergera. Hur många generationer som krävs har vi försökt utreda genom att exekvera en testfunktion ett stort antal gånger med varierande kromosomlängd. För att undvika att blanda in begrepp som upplösning och för att få en snabb och enkel funktion att optimera skapade vi en biträknande funktion. Godhetstalet är alltså lika med antalet ettor i kromosomen. Detta val av testfunktion har även fördelen att problemets storlek är ekvivalent med kromosomlängden samt att det optimala värdet är känt för varje kromosomlängd.

Vi har låtit GA konvergera till 95 procent respektive 99 procent av optimum i de båda testserierna. Vi har låtit medelvärdesbilda antalet generationer och plottat dem mot kromosomlängden i figur 3.9 nedan. Resultat tyder på ett ganska linjärt samband mellan kromosomlängd och antalet generationer vid konvergens upp till 95 procent av optimum. För konvergens upp till 99 procent av optimum är sambandet snarare kvadratiskt.



Figur 3.9 Antal generationer som funktion av kromosomlängden.

Tidskomplexitet blir då T(n) = $O(n^2)$ respektive T(n) = $O(n^3)$. För att verifiera detta har vi också plottat den medelvärdesbildade tidsåtgången mot kromosomlängden för de olika testserierna i figur 3.10. Notera särskilt hur tiden för att konvergera från 95 procent av optimum till 99 procent exploderar med ökande kromosomlängd.



Figur 3.10 Tid för 95% resp. 99% konvergens som funktion av kromosomlängden

Under alla körningar av testfunktionen har vi hållit populationstorleken fix vid en relativt hög nivå, det bör också nämnas att testfunktionen saknar beroende inom kromosomen.

Det finns även ett samband mellan kromosomlängden och populationstorleken, d v s en viss kromosomlängd har en viss optimal populationsstorlek. Detta har vi dock inte utrett vidare. Vi vill också understryka att vi ej testat konvergensen för kromosomer med beroende, resultatet ovan gäller alltså ej dessa kromosomer. Det enda vi kan säga med säkerhet är att dessa kromosomer konvergerar långsammare, hur mycket sämre beror på styrkan av beroendet.

3.4 SAMMANFATTNING

Vi har i detta kapitel diskuterat olika inställningar, funktioner och deras effekter på GA-körningar. Slutmålet är naturligtvis att få GA att konvergera så snabbt som möjligt.

En de mer framstående forskarna inom GA-området, David E Goldberg, framhåller i en artikel[10] vikten av att hålla de genetiska algoritmerna så nära naturen som möjligt och undvika all styrning genom överordnade regler. Genom att låta "naturen" ha sin gång med hjälp av sunda, förnuftiga parametrar uppnår man ett överlägset resultat. Detta överensstämmer med våra resultat och erfarenheter.

En av de upptäckter som förvånade oss mest när vi jobbade med att öka konvergenshastigheten var upplösningens betydelse. Konvergenshastigheten ökade betydligt då vi ökade upplösningen på problemets variabler. I övrigt kom vi fram till att man kan öka konvergenshastigheten betydligt genom att ställa inparametrarna rätt. Att ta fram de bästa inparametrarna är dock ett mycket tidsödande jobb.

Till de mer negativa upptäckterna hörde problemen med inbyggda beroenden i kromosomen. Dessa beroenden kan ofta inte undvikas med följden att konvergens-hastigheten blir lidande. En annan negativ upptäckt var naturligtvis algoritm-komplexiteten som visar att stora problem, som inte kan delas upp i mindre del-problem, är olösliga med hjälp av GA.

[10]Goldberg David E, Zen and the Art of Genetic Algorithms, Proceedings of the Third International Conference of Genetic Algorithms, Morgan Kaufmann Publishers, 1989

4. OPTIMERING AV ETT LINJÄRT KRAFTSYSTEM

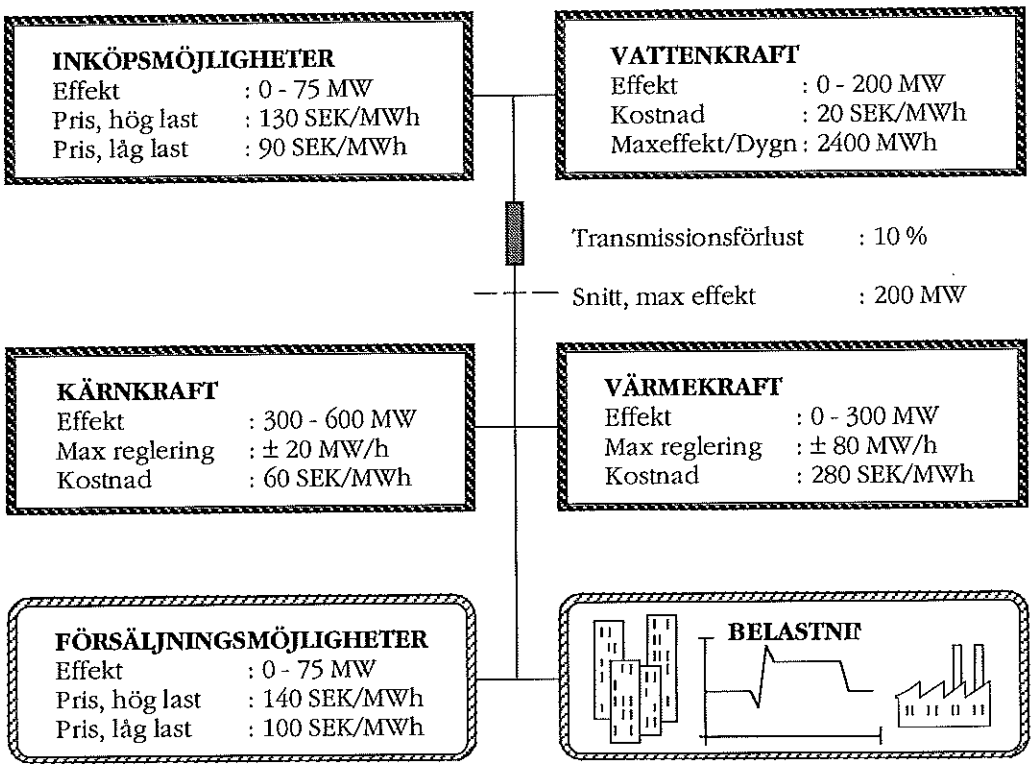
För att utvärdera GA:s beteende på produktionsoptimeringsproblem skapade vi nedanstående kraftsystem. Målet med optimeringen var att minimera den totala rörliga kostnaden för att producera eller köpa in den kraft[11] som krävdes för att möta efterfrågan. Efterfrågan representerades av den prognosticerade belastningskurvan.

Syftet med modellen har inte varit att skapa en bild av Sydkrafts produktionsapparat utan att möjliggöra en granskning av GA:s prestanda på denna typ av problem. Systemet innehåller modeller över de väsentligaste typerna av kraftproducerande och kraftkonsumerande objekt.

För att vi skulle kunna analysera GA:s resultat krävdes en alternativ optimeringsmetod. Vårt val blev linjär programmering [LP], då det är en mycket robust optimeringsmetod som dessutom klarar av att hantera mycket stora problem. Vi valde en tidshorisont om ett dygn med timupplösning för optimeringen. Detta för att erhålla rimliga beräkningstider på problemet.

4.1 MODELL

Kraftsystemet består av fyra objekt som matar in kraft på nätet, ett vattenkraftverk, ett kärnkraftverk, ett värmekraftverk, samt ett affärobjekt som möjliggör inköp av kraft. Modellen har dessutom två objekt som förbrukar kraft, en aggregerad last bestående av kraftbolagets alla kunder samt ett objekt som möjliggör försäljning av överskottskraft. Vårt enkla kraftsystemet är uppbyggt som figur 4.1. visar.



Figur 4.1 Kraftsystemet

[11]Här används begreppet "Kraft" istället för det korrektare "Energi" eftersom det är det gängse begreppet i branschen.

Transmission av kraft mellan kraftsystemets norra och södra del innebär en 10-procentig kraftförlust. Ytterligare finns en fysisk begränsning på den överförda kraften vid 200 MW. För att systemet skall balansera krävs att inmatad kraft på nätet är lika med den uttagna kraften.

4.1.1 VÄRMEKRAFT

Vi har varit tvungna att förenkla värmekraftverket för att kunna transformera modellen till ett LP-problem. Detta har inneburit att vi varit tvungna att slopa uppstartskostnaderna. Vi kan inte heller formulera regler för grundkraft, d v s att man kan köra ett värmekraftverk i intervallet 80-320 MW eller inte alls. Nivån 80 MW är då värmekraftverkets grundkraft. Vi bortser även här från eventuella kostnader för upp- resp. nedregleringen. Nedanstående beskriver värmekraftverket:

Max effekt	300 MW
Min effekt	0 MW
Max reglering	± 80 MW/h
Rörlig kostnad	280 kr/MWh

4.1.2 KÄRNKRAFT

Vi har i vår modell inte tagit med möjligheten att stänga av kärnkraftverket eftersom det oftast är ett problem där man tar hänsyn till fler faktorer än bara produktionskostnaden. Vår modell över kärnkraftverket beskrivs av följande:

Max effekt	600 MW
Min effekt	300 MW
Max reglering	± 20 MW/h
Rörlig kostnad	60 kr/MWh

4.1.3 INKÖP

Detta objekt möjliggör inköp av extern kraft. Modellen arbetar med två olika priser för kraften beroende på tidpunkt för köpet. Högbelastningstid i modellen är mellan 07.00 och 22.00, resten är lågbelastningstid. Inköpen får variera maximalt mellan timmarna, d v s man får köpa max effekt vid timme t för att timme t+1 inte köpa någon kraft alls och vice versa. Följande gäller för objektet:

Max effekt	75 MW
Min effekt	0 MW
Pris, hög last	130 SEK/MWh
Pris, låg last	90 SEK/MWh

4.1.4 FÖRSÄLJNING

Man har även möjlighet att sälja kraft i modellen. På samma sätt som vid inköp gäller två olika prisnivåer. Försäljningen får variera maximalt mellan timmarna. Följande gäller för försäljningen:

Max effekt	75 MW
Min effekt	0 MW
Pris, hög last	140 SEK/MWh
Pris, låg last	100 SEK/MWh

4.1.5 VATTENKRAFT

Vi har skapat ett förenklat vattensystem genom att införa ett vattenmagasin med tillhörande turbin. Till magasinet är kopplat dämningsgränser. Vi har vidare bestämt att 1 volymenhet [v.e] vatten är ekivalent med den mängd vatten som ger 1 MWh energi.

Reglerna för vattenkraftsobjektet är enligt nedanstående tabell:

Max tappning	200	v.e/h
Min tappning	0	v.e/h
Mellantillrinning	100	v.e/h
Ingående magasinsläge	10000	v.e
Övre dämningsgräns	11000	v.e
Undre dämningsgräns	9000	v.e
Rörlig kostnad	20	SEK/MWh

Slutläget av magasinet skall efter ett dygn vara lika med det ingående magasinsläget. Vattenkraften kan regleras upp/ned hur mycket som helst.

4.1.6 BELASTNING

Kraftsystemet skall betjäna ett stort antal konsumenter av olika storlekar och konsumtionsmönster. Deras aggregerade efterfrågan representerar av nedanstående belastningskurva. Modellens belastningskurva är vald så att den efterliknar en normal veckodags belastning. Man ser tydligt toppen runt sju på morgonen när industrin startar upp. Vidare ser man även att den lägsta belastningen är placerad under nattens sena timmar. Nivån på belastningen är anpasssad så att vi ska kunna få ut så mycket information som möjligt ur vår optimering.

Figur 4.2 Belastningskurva.

4.2 IMPLEMENTERING

För att kunna göra en LP-implementering av modellen har vi varit tvungna att införa vissa förenklingar redan på modellstadiet. Den allvarligaste är att vi varit tvungna att slopa uppstartskostnaderna för värmekraftverket. Detta kompenseras dock av vårt val av belastningskurva som tvingar till uppstart av värmekraftverket oavsett kostnaden. Vi har inte heller kunnat formulera regler för grundkraft hos värmekraftverket eller använda oss av olinjära priskurvor vid försäljning och inköp.

4.2.1 LP-IMPLEMENTERING

Vi gjorde vår första implementering av LP-modellen i MatLab, och använde oss av de färdigskrivna rutinerna i Optimization_Toolbox. Dessvärre visade det sig att MatLab inte klarade av att hantera problem av denna storlek. Vi försökte då med programmet lp_solve som visade sig fungera utmärkt.

Vår LP-modell bestod av totalt 140 variabler och 383 begränsningar. Våra variabler bestod av fem vektorer heat[i], nuclear[i], buy[i], sell[i] och water[i]. Vardera vektorn är 24 element lång, en för varje timme under optimeringsperioden.

Kostnadsfunktionen vars värde skulle minimeras formulerades enligt nedan:

$$\sum_{i=0}^{23} (280 \cdot heat[i] + 60 \cdot nuclear[i] + buyprice[i] \cdot buy[i] + sellprice[i] \cdot sell[i] + 20 \cdot water[i])$$

Värden för inköpspris, Buyprice, och försäljningspris, Sellprice, ges i tabellen nedan:

Buyprice[i]		Sellprice[i]	
130 SEK/MWh	i = 7,8 ... 21	140 SEK/MWh	i = 7,8 ... 21
90 SEK/MWh	övriga	100 SEK/MWh	övriga

Problemets begränsningar ges nedan:

Tillförd effekt = konsumerad effekt

heat[i] + nuclear[i] + 0.9·buy[i] + 0.9·water[i] - sell[i] = belastning[i]
i = 0... 23

Max-effekt begränsningar

heat[i] <= 300 MW
i = 0... 23

nuclear[i] <= 600 MW
i = 0... 23

buy[i] <= 75 MW
i = 0... 23

sell[i] <= 75 MW
i = 0... 23

water[i] >= 0 MW
i = 0... 23

Min-effekt begränsningar

heat[i] = 0 MW /* Värmekraft */
i = 0... 23

nuclear[i] >= 300 MW /* Kärnkraft */
i = 0... 23

buy[i] >= 0 MW /* Inköp */
i = 0... 23

sell[i] >= 0 MW /* Försäljning */
i = 0... 23

Transmissionsbegränsning

0.9·buy[i] + 0.9·water[i] <= 200 MW
i = 1... 23

Nedregleringsbegränsning

nuclear[i]-nuclear[i+1] <= 20 MW
i = 0... 22

heat[i]-heat[i+1] <= 80
i = 0... 22

Uppregleringsbegränsning

nuclear[i]-nuclear[i+1] >= -20 MW
i = 0... 22 /* Kärnkraft */

heat[i]-heat[i+1] >= -80 MW
i = 0... 22 /* Värmekraft */

Begränsning av totalt uttagen vattenkraft

$$\sum_{i=0}^{23} water[i] <= 2400 \text{ volymenheter (MWh)}$$

Övre vattendom

$$\sum_{k=0}^{i} water[k] - 100 \cdot k <= 1000 \text{ volymenhet.}$$

i = 11... 23

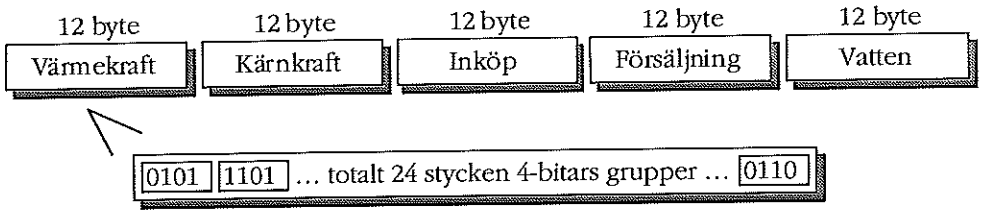
Undre vattendom

$$\sum_{k=0}^{i} water[k] - 100 \cdot k >= 1000 \text{ volymenhet.}$$

i = 11... 23

4.2.2 GA-IMPLEMENTERING 1

Den första implementeringen vi valde att prova bestod av en kromosom som var uppdelad i fem delar, en del vardera för värmekraftverket, kärnkraftverket, försäljningen, inköp och vatten. Varje del bestod i sin tur av 12 bytes, d v s 24 stycken 4-bitars tal. Se figur 4.3 nedan. Avkodningen av kromosomen skedde så att första 4-bitars gruppen i värmekraftdelen beskrev körningsnivå timme 0, andra 4-bitars gruppen beskrev körningsnivå timme 1 osv.

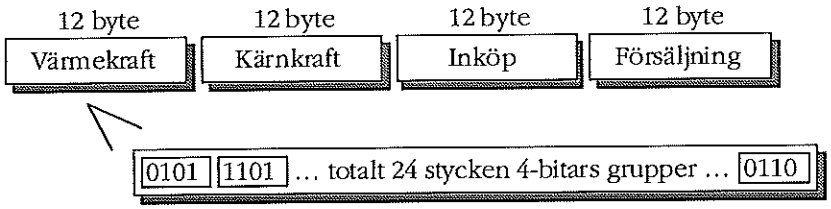


Figur 4.3 Kromosomstrukturen för vattenimplementeringen.

Ett problemet som uppstod var att GA inte producerar exakt den mängd energi som konsumeras. Detta löste vi med att straffa kromosomen om den producerade för lite och låta den straffa sig själv med den ökade kostnad som uppstår vid överproduktion.

4.2.3 GA-IMPLEMENTERING 2

Resultatet visade sig inte bli riktigt bra med ovanstående implementering, så vi valde att prova med att ta ut vattenkraften ur kromosomen. Vattenkraften räknade vi sedan ut genom beräkna skillnaden mellan tillförd och bortförd kraft vid belastningsnoden. Alltså såg vi till att täcka återstoden av belastningen med hjälp av vattenkraften. Detta gjorde att GA alltid producerade lika mycket som skulle konsumeras och vi fick bättre lösningar. Kromosomstrukturen blev då som figur 4.4 visar.



Figur 4.4 Kromosomstrukturen som gav bäst resultat.

Vi använde oss sedan av vårt egenkonstruerade C-program för att optimera problemet. De bästa körningsresultaten uppnåddes då vi använde oss av uniform överkorsning och linjärt avtagande mutationssannolikhet. Det visade sig att inställningen av parametrar är kritisk och vi lade ner mycket tid i provkörningar för att bestämma de bästa parametervärdena. Den i särklass viktigaste enskilda parametern är sannolikheten för mutation. Vi valde den till 0.006 i vårt exempel men det optimala värdet beror på problemets karaktär och kromosomlängd.

4.3 RESULTAT

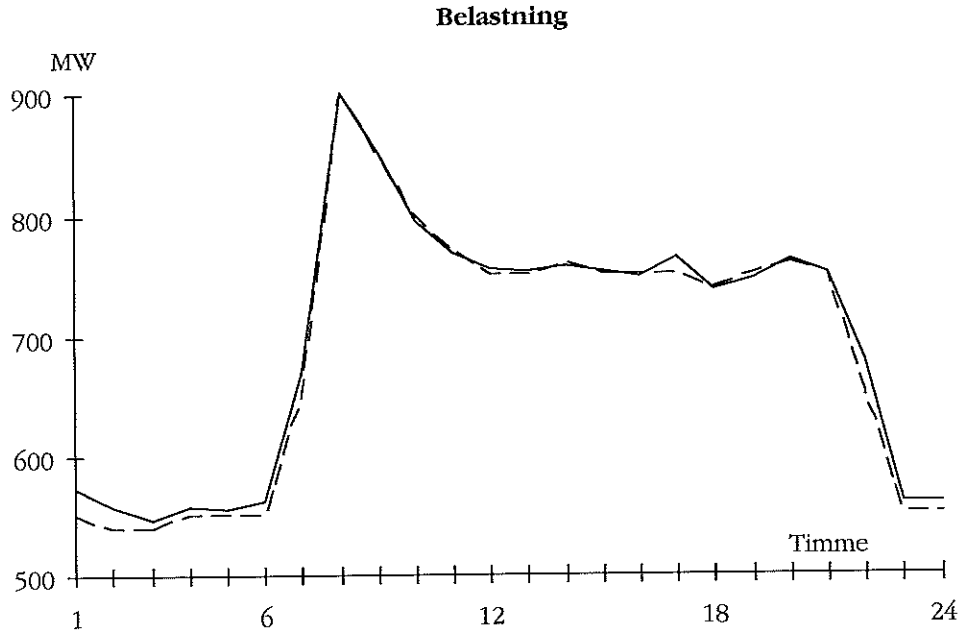
I detta avsnitt presenterar vi de två bästa lösningarna framtagna med respektive kromosomstruktur. Vi jämför lösningarna med den kända optimala lösningen vi tagit fram med hjälp av LP. Resultatet för implementering 2 är tillfredsställande och överensstämmer med våra förväntningar. Implementering 1 visar sig vara betydligt

sämre. Värt att notera är även exekveringstiden för de olika metoderna, GA tar ungefär en och en halv timme på en Macintosh IIfx medan LP tar bortåt en halv sekund på en DECstation 5000.

4.3.1 RESULTAT AV GA-IMPLEMENTERING 1

Resultatet av den bästa körningen med GA-implementering 1, d v s implementeringen där vattenkraften är med i kromosomen, visas nedan. Körplanerna visas i figur 4.6. Figur 4.5 visar en förstoring av belastningskurvan och GA:s totala produktion och inköp.

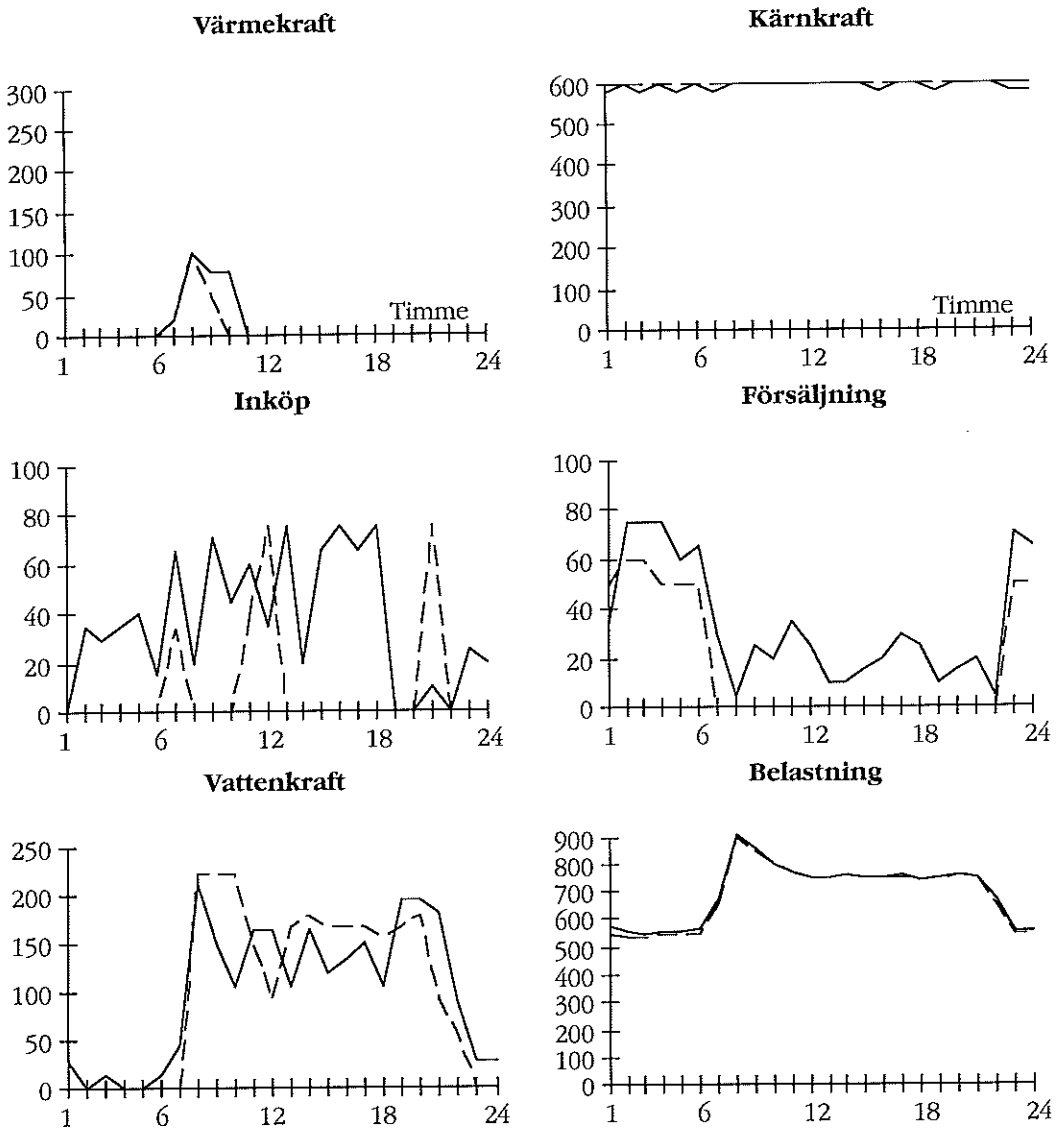
```
NO GRAY - Lastkurva 2 - Genetiska parametrar:

Sannolikhet för överkorsning : 0.850000                    Uniform överkorsning
Sannolikhet för mutation     : 0.006000 Linjärt avtagande mutationssannolikhet
Sannolikhet för migration    : 0.120000                    Konstant straffunktion

Antal delpopulationer        :    10     Antal individer i varje     :     20
Antal generationer           : 10000     Kromosomlängd i bytes       :     60
Filnamn                      : apa

Gen:     0  Max: 7.9030e+06  Avr: 7.435e+06  CO:      0  MU:       0  MI:      0
Gen:  1000  Max: 8.7949e+06  Avr: 8.420e+06  CO:  84956  MU:  553498  MI:  23223
Gen:  2000  Max: 8.7971e+06  Avr: 8.507e+06  CO: 169898  MU: 1060350  MI:  46459
Gen:  3000  Max: 8.8050e+06  Avr: 8.592e+06  CO: 254894  MU: 1520887  MI:  69985
Gen:  4000  Max: 8.8333e+06  Avr: 8.562e+06  CO: 339849  MU: 1935585  MI:  93709
Gen:  5000  Max: 8.8596e+06  Avr: 8.655e+06  CO: 424901  MU: 2303557  MI: 116630
Gen:  6000  Max: 8.9087e+06  Avr: 8.736e+06  CO: 509802  MU: 2626861  MI: 140745
Gen:  7000  Max: 8.9130e+06  Avr: 8.762e+06  CO: 594671  MU: 2903467  MI: 164470
Gen:  8000  Max: 8.9432e+06  Avr: 8.844e+06  CO: 679458  MU: 3133255  MI: 187972
Gen:  9000  Max: 8.9660e+06  Avr: 8.890e+06  CO: 764483  MU: 3317234  MI: 211501
Gen: 10000  Max: 8.9930e+06  Avr: 8.926e+06  CO: 849457  MU: 3455054  MI: 234533

Current time: 11:13:16 - 1993.07.07,  Executiontime: 1 h, 20 min, 12 sek
```



Figur 4.5 Belastningskurvan (streckad) och GA:s totala produktion (hel).

Man ser i figur 4.5 att GA:s produktion inte är lika med förbrukningen, utan det finns hela tiden ett glapp kurvorna emellan. Man kan ha överseende med detta om GA bara håller sig inom ett visst givet intervall runt belastningskurvan och att totalproduktionen

under alla 24 timmarna blir ungefär korrekt. Anledningen är att prognosen för belastningskurvan man skall optimera är ganska grov.



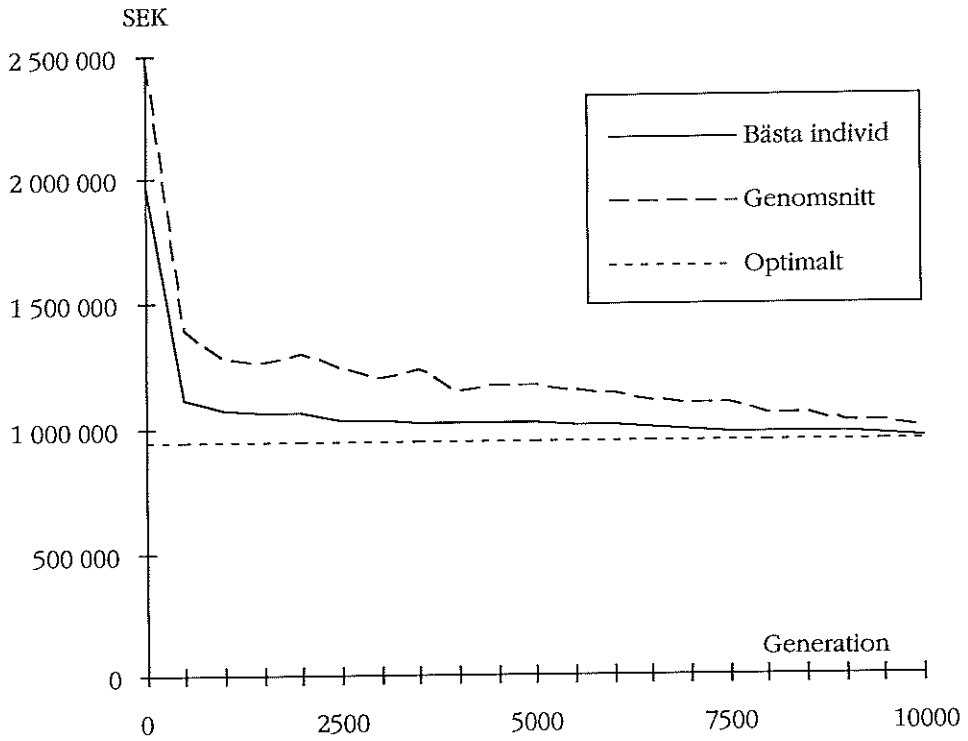
Figur 4.6 Körplaner enligt GA (heldragen) och LP (streckad).

Bästa individens lösning blev till slut en totalkostnad på 1 007 000 SEK eller 6.5 procent högre än den optimala. Detta resultat är betydligt sämre än resultatet för implementering 2. Detta resultat beskrivs i nästa avsnitt.

4.3.2 RESULTAT AV GA-IMPLEMENTERING 2

Nedan redovisar vi vår framgångsrikaste GA-körning med implementering 2, d v s implementeringen där vi räknar fram vattenkraftens körningsplan. Bästa individens totalkostnad och populationens genomsnittliga totalkostnad visas i figur 4.7 nedan. Vi kan se hur populationen relativt snabbt konvergerar mot en förhållandevis låg totalkostnad. Därefter sker endast smärre steg mot den lägsta möjliga kostnaden med hjälp av lyckosamma mutationer och överkorsningar.

```
NO GRAY - Lastkurva 2 - Genetiska parametrar:

Sannolikhet för överkorsning : 0.850000                    Uniform överkorsning
Sannolikhet för mutation     : 0.006000 Linjärt avtagande mutationssannolikhet
Sannolikhet för migration    : 0.120000                    Konstant straffunktion

Antal delpopulationer        :    10      Antal individer i varje      :     20
Antal generationer           : 10000      Kromosomlängd i bytes        :     48
Filnamn                      : tst21.16

Gen:     0  Max: 8.0354e+06  Avr: 7.531e+06  CO:      0  MU:      0  MI:      0
Gen: 1000   Max: 8.9267e+06  Avr: 8.723e+06  CO:  85129  MU: 441941  MI:  23774
Gen: 2000   Max: 8.9389e+06  Avr: 8.700e+06  CO: 170322  MU: 846805  MI:  47141
Gen: 3000   Max: 8.9694e+06  Avr: 8.800e+06  CO: 255329  MU:1215608  MI:  70455
Gen: 4000   Max: 8.9754e+06  Avr: 8.848e+06  CO: 340274  MU:1546867  MI:  94042
Gen: 5000   Max: 8.9828e+06  Avr: 8.826e+06  CO: 425367  MU:1841783  MI: 116893
Gen: 6000   Max: 8.9952e+06  Avr: 8.863e+06  CO: 510203  MU:2099587  MI: 140909
Gen: 7000   Max: 9.0156e+06  Avr: 8.903e+06  CO: 594931  MU:2320697  MI: 164326
Gen: 8000   Max: 9.0262e+06  Avr: 8.946e+06  CO: 679867  MU:2505060  MI: 187405
Gen: 9000   Max: 9.0268e+06  Avr: 8.978e+06  CO: 764939  MU:2651388  MI: 210571
Gen:10000   Max: 9.0437e+06  Avr: 9.008e+06  CO: 849994  MU:2761590  MI: 234378

Current time: 13:32:55 - 1993.07.07,  Executiontime: 1 h, 38 min, 27 sek
```

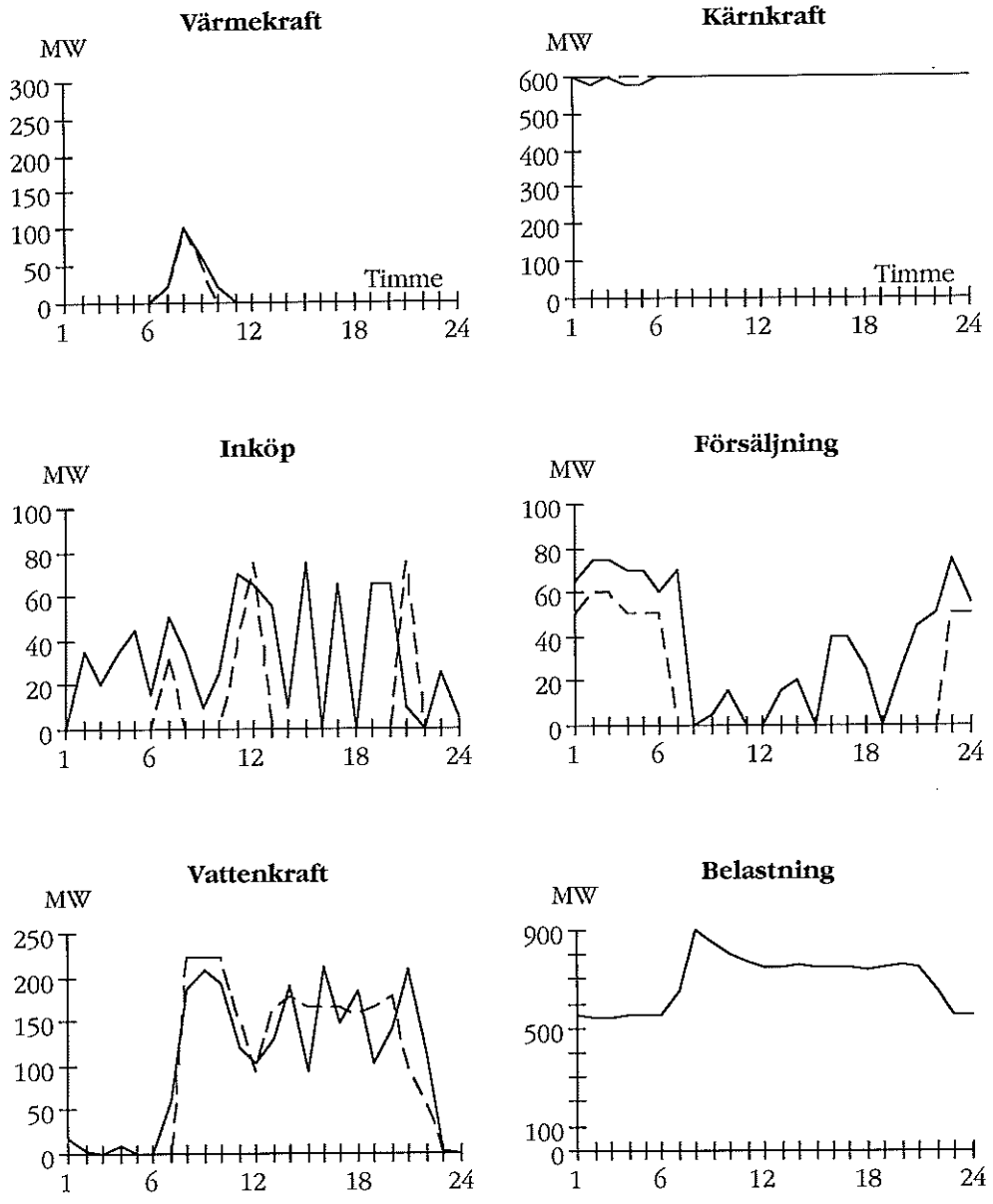


Figur 4.7 GA-resultat som funktion av antalet generationer.

Diagrammen i figur 4.8 nedan visar hur produktion, inköp och försäljning skall ske i respektive objekt för att uppnå en minimal totalkostnad. LP-lösningen, som är den optimala representeras av de streckade linjerna. GA-resultatet visas av de heldragna linjerna.

Om vi tittar närmre på GA-resultatet ser vi att värmekraftverket, som har den högsta marginalkostnaden, endast kör under belastningstoppen. Detta inträffar under tiden 06.00 - 09.59, därefter används verket ej. Kärnkraftverket som har låg marginalkostnad och är svårreglerad, kör för fullt nästan hela optimeringsperioden. När vi betraktar

inköps- och försäljningskurvorna är GA-resultatet i bästa fall brusigt. Vi tror att det beror på att kostnadsskillnaden mellan att köpa in för mycket kraft och sälja av den är försumbar om man inte överträder snittbegränsningen. Vattenkraften slutligen producerar kraft under dagtid då belastningen är hög, medan den spar vatten under nattens lågbelastningstimmar.



Figur 4.8 Körplaner enligt GA (heldragen) och LP (streckad).

Om man jämför GA-resultatet med Sydkrafts nuvarande manuella optimering finner man stora likheter. Värmekraftverk används endast i nödfall för att möta en hög belastning på grund av den höga produktionskostnaden. Även vattenkraftkörningen uppvisar likheter, man tappar vatten under dagtid och spar nattetid. Sydkraft använder även kärnkraften som bas, kärnkraften går med konstant effekt under mycket långa perioder i sträck. När det däremot gäller handel med kraft är likheterna inte lika slående. I dessa fall uppvisar GA ett brusdränkt resultat.

GA gav en minimal totalkostnad på 956300 SEK mot den optimala lösningen på 945160 SEK. Skillnaden är mindre än 1.2 procent. Det bör noteras att GA arbetar med diskreta körningssteg om ±20 MW för kärnkraft och värmekraft och ±5 MW för handel. Därmed har GA inte samma möjlighet att finjustera körningarna som LP kan göra.

4.4 SAMMANFATTNING

I detta kapitel har vi undersökt GA:s beteende på ett linjärt kraftsystem. Vi har tagit fram systemets optimala lösning med hjälp av LP. Därefter har vi optimerat systemet med två olika kromosomkodningar. Resultatet är lovande, med den bästa implementeringen når man mycket bra lösningar, endast 1.2 procent från optimum. Tyvärr är GA väldigt mycket långsammare än LP. GA har dock fördelen att de klarar av optimering av olinjära kraftsystem.

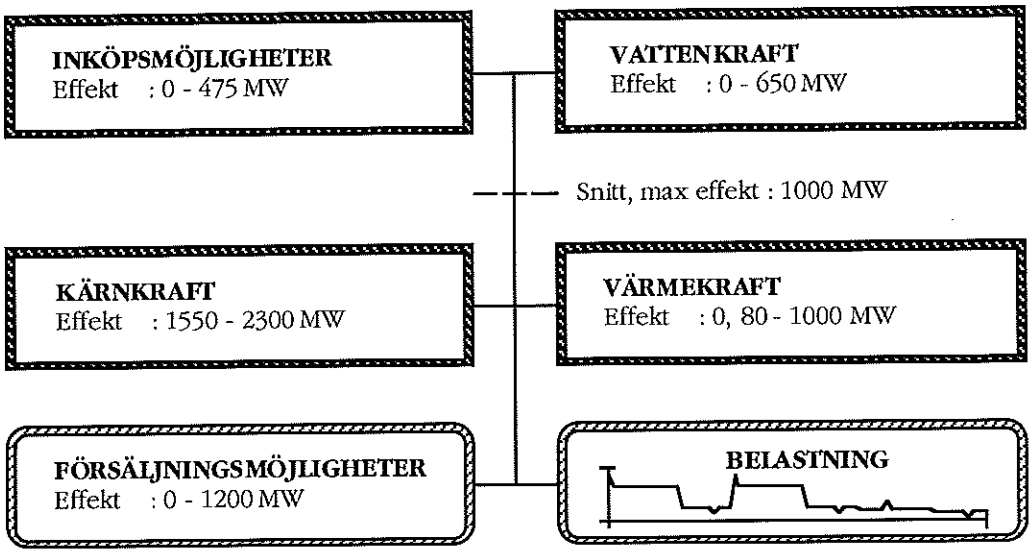
Intressant är att både LP- och GA-lösningen visar stora likheter med den manuella optimering Sydkraft använder idag. Detta är ett godtagbart resultat men förhoppningen är att man ska finna bättre lösningar än man gör manuellt då problemet i verkligheten innehåller en mängd olinjäriteter som är svåra att överblicka. I nästa kapitel ska vi undersöka om GA även klarar av ett olinjärt problem lika galant som vårt linjära.

5. OPTIMERING AV ETT OLINJÄRT KRAFTSYSTEM

Efter att ha analyserat GA:s beteende och lösning av linjära produktionsoptimeringsproblem, ville vi undersöka GA även på olinjära problem. Vi valde en tidshorisont på tre dygn. Valet av tre dygn var en kompromiss mellan tidsaspekt och verklighetsanknytning. De tre dygnen bestod av två vardagar och en helgdag av den anledningen att vi ville testa om GA skulle göra på samma sätt som Sydkraft gör idag, d v s spara vattnet under veckoslutet för att kunna använda det fullt ut under vardagarna. Att vi inte valde en hel vecka, berodde på att redan tre-dygns optimeringen tog cirka 36 timmar att lösa på en Macintosh Quadra 700.

5.1 MODELL

Kraftsystemet bestod även i denna lite större modell av sex objekt, fyra som producerar och två som konsumerar elkraft. Det som skiljer denna modell från den förra, är att vi här har introducerat flera olinjäriteter och ökat upplösningen. Anledningen till att vi ökade upplösningen till åtta bitar var dels att konvergenshastigheten ökade, se kapitel 3, och dels att det var för grovt att dela in till exempel inköpens 800 MW i steg om 50 MW. Ändringarna vi gjorde resulterade i att man kan betrakta modellen som en mycket grov modell över Sydkrafts produktionsapparat.



Figur 5.1 Kraftsystemet

Figur 5.1 ovan visar kraftsystemets uppbyggnad. En skillnad från kapitel 4:s kraftsystem är att vi reducerat bort transmissionsförlusten genom att höja inköpspriserna och justera vattenkraftekvivalenten i motsvarande grad. Vattenkraftekvivalenten är den mängd energi man får vid tappning av en viss mängd vatten. I vår modell är vattenkraftekvivalenten 0.67, d v s vid tappning av 980 m^3/s erhåller man 980·0.67 = 650 MW.

5.1.1 VÄRMEKRAFT

Det värmekraftverk som vi har valt att modellera, är en grov förenkling av Karlshamnsverket. I vår modell har vi tagit hänsyn till uppstartskostnaderna genom att inkludera dessa i priset på värmekraften. Vi har även formulerat regler för grundkraft, d v s att man kan köra värmekraftverket antingen med effekten 0 MW eller i intervallet 80 till 1000 MW. Vi har även infört begränsningar på upp och nedregleringar, men vi har bortsett från eventuella kostnader för dessa. Vidare har vi slagit samman

Karlshamnsverkets tre aggregat till ett enda. Nedanstående siffror beskriver värmekraftverket:

Max effekt	1000 MW
Min effekt	0 MW
Grundkraft	80 MW
Max reglering	± 240 MW/h
Rörlig kostnad	280 kr/MWh

5.1.2 KÄRNKRAFT

Sydkraft äger Barsebäcksverket med två reaktorer på vardera 600 MW. Dessutom äger man cirka 50 procent av Oskarshamns Kraftgrupp AB, och därmed lika stor andel av Oskarshamnsverket. Genom avtal disponerar man 1100 MW av den totala nettoeffekten på 2215 MW. Vi har, i vår modell, inte tagit med möjligheten att stänga av kärnkraftverket, eftersom det oftast är ett problem där man tar hänsyn till fler faktorer än bara produktionskostnaden. Däremot har vi tagit hänsyn till de upp- och nedregleringsbegränsningar som finns. Vår modell, som är en sammanslagning av Barsebäck och Oskarshamnsverket till ett kärnkraftverk, beskrivs av följande:

Max effekt	2300 MW
Min effekt	1550 MW
Max reglering	± 100 MW/h
Rörlig kostnad	60 kr/MWh

5.1.3 INKÖP

Sydkrafts inköp på kraftbörsen, främst från Norge, svarar för cirka 10 procent av Sydkrafts totala kraftproduktion. Utbyte av el med Norge var tänkt att modelleras med hjälp av ett neuralt nätverk, som ger en prognos av utbytespriserna för en vecka framåt i tiden. Då detta nätverk inte fungerade speciellt bra och den godhetsfunktion som genererades av nätverket tog alldeles för lång tid, fick vi göra en grov uppskattning istället. Modellen arbetar med olika priser för kraften beroende på tidpunkt och kvantiteten för köpet. Högbelastningstid i modellen är mellan 07.00 till 21.00, resten är lågbelastningstid. Inköpen får variera maximalt mellan timmarna, d v s man får köpa max effekt vid timme t för att timme t+1 inte köpa någon kraft alls och vice versa. Följande gäller:

Höglast (07-21)		Låglast (21-07)	
Pris 0 - 200 MW	200 SEK/MWh	Pris 0 - 200 MW	160 SEK/MWh
Pris 200 - 400 MW	140 SEK/MWh	Pris 200 - 400 MW	115 SEK/MWh
Pris 400 - 475 MW	120 SEK/MWh	Pris 400 - 450 MW	90 SEK/MWh

5.1.4 FÖRSÄLJNING

Sydkraft säljer elkraft till i första hand Danmark. Det vi ville undersöka var om GA tänkte rätt, d v s att man säljer så fort man kan göra vinst och har möjlighet att sälja. På samma sätt som vid inköp gäller olika prisnivåer beroende på tid på dygnet och såld kvantitet. Försäljningen får variera maximalt mellan timmarna. Följande gäller:

Höglast (07-21)		Låglast (21-07)	
Pris 0 - 100 MW	160 SEK/MWh	Pris 0 - 100 MW	112 SEK/MWh
Pris 100 - 200 MW	130 SEK/MWh	Pris 100 - 300 MW	90 SEK/MWh
Pris 200 - 1100 MW	50 SEK/MWh	Pris 300 - 1200 MW	50 SEK/MWh

5.1.5 VATTENKRAFT

Sydkraft äger intressen i ett 70-tal vattenkraftverk i Sydsverige och ett 20-tal vattenkraftverk i Norrland. Vattenkraften svarar för cirka en tredjedel av den totala produktionen, där de norrländska älvarna står för cirka 90 procent av kraften. Vi har

skapat ett förenklat vattensystem genom att införa ett vattenmagasin med tillhörande turbin och placerat detta i Norrland. Till magasinet är kopplat dämningsgränser. Systemets regler är följande:

Max tappning	980	t.e
Min tappning	0	t.e
Mellantillrinning	365	t.e
Kraftekvivalent	0,67	MW/t.e
Rörlig kostnad	20	SEK/MWh

Tappnings- och tillrinningsvolymer anges ovan i timenheter, t.e, vilket motsvarar volymen av 1 m^3/s i en timme, d v s 3600 m^3. Uteffekten beräknas genom att multiplicera tappningsvolymen med kraftekvivalenten. Slutläget av magasinet skall vara lika med ingående magasinsläge efter tre dygn, d v s vi får inte ta ut mer än 72·365 t.e vatten på de tre dygnen. Vattenkraften kan regleras upp/ner hur mycket som helst.

5.1.6 ÖVRIGT

Sydkraft har även andra kraftproducerande objekt än de ovanstående, framförallt gasturbiner. Gasturbiner skulle egentligen vara ganska enkla att implementera, dels för att deras kostnadsfunktion är mycket enkel, dels för att de har ett mycket snabbt uppstartsförlopp vilket gör att man inte behöver bry sig om upp- och nedregleringsbegränsningar. Gasturbinernas stora nackdel är att de kostar väldigt mycket och att de har en ganska begränsad effekt, totalt ca 350 MW. Det är anledningen till att vi inte tog med dem i vår modell.

Sydkraft har också möjlighet att ändra sin belastning. Det sker främst genom avbrytbara leveranser i form av elpannor. Sydkraft kan alltså minska sin last genom att stänga av sina kunders elpannor och på det sättet undvika dyra uppstarter av värmekraftverk. Sydkraft har kunder med så kallade 3000 och 8000 timmars abonnemang, där Sydkraft garanterar att leverera kraft under 3000 respektive 8000 timmar under ett år. Den övriga tiden kan Sydkraft, om man så önskar, stänga av pannorna. Kraften kan stängas av momentant, d v s nedregleringstiden är noll. Det finns dock olika avtal som innebär att Sydkraft måste informera om bortkoppling, ett visst antal timmar tidigare. Avkopplingstid är en annan parameter, som också regleras i dessa avtal. Trenden är att i framtiden kommer avtalen att bli mer individuellt anpassade och skräddarsydda efter kundens önskemål. Detta försvårar sammanslagning av elpannorna till ett gemensamt objekt.

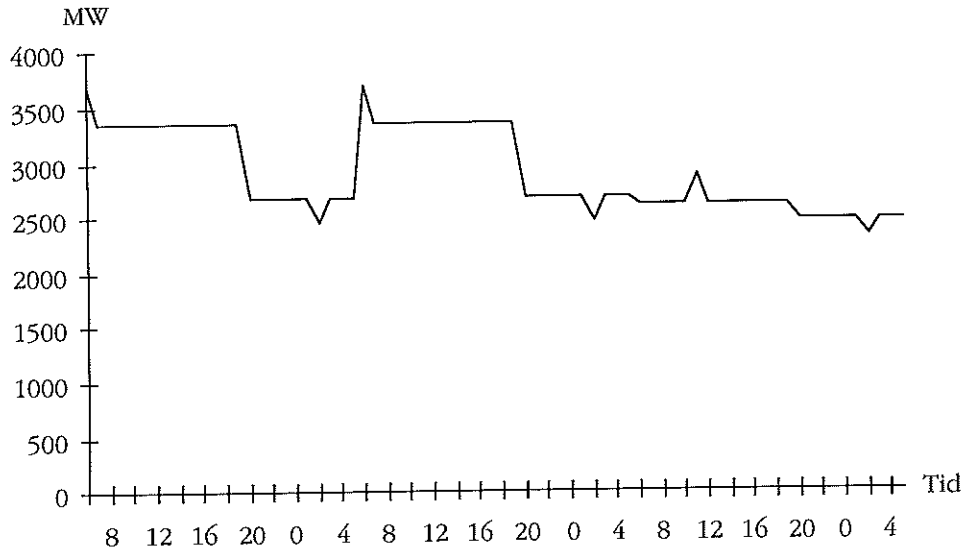
En annan svårighet är att man måste anlägga ett ett-års perspektiv för avkopplingen av elpannor, eftersom kortsiktigt ökar vinsten vid en nedreglering. Detta innebär att man inte kan låta de genetiska algoritmerna bestämma avkopplingen, eftersom de i så fall bara kommer att koppla ur samtliga elpannor för att minimera kostnaden. Detta är huvudanledningen till att vi strök dessa ur modellen.

5.1.7 BELASTNING

Belastningskurvan under optimeringsperioden har vi tagit fram med hjälp av data från Sydkraft. Som man ser i bild 5.2 är varje dygn indelad i en hög- och en lågbelastningsperiod. I respektive period finns det ett extremvärde. Högbelastningsperiodens extremvärde för vardagar är en topp kl sju på morgonen då industrin startar upp. Samma topp under helgdagen ligger runt klockan ett då folk gör sin söndagsmiddag. För lågbelastningstimmarna finns en dal som representerar den lägsta förbrukningen på dygnet. Denna timme brukar inträffa någon gång mellan fyra och fem på morgonen.

I vårt fall var belastningen given för hela perioden. I verkligheten används flera olika prognosticeringsmetoder för att förutsäga den kommande veckans förbrukning. Man använder tidserieanalys, neurala nätverk och kunskap från tidigare veckor med likartad vädersituation och belastningsmönster. Prognosmetoder ger alltid ett visst fel och man kan därför tillåta att GA kör kraftverken fel, om felet är begränsat.

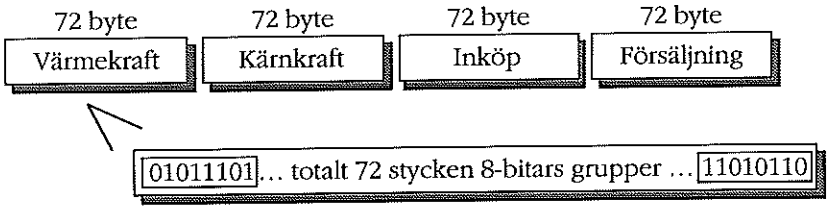
I vår modell antar vi att belastningen är fullständigt känd och därmed tolererar vi inga felkörningar.



Figur 5.2 Belastningskurvan för vår tredygnsoptimering

5.2 GA-IMPLEMENTERING

Implementeringen vi valde, var en kromosom som likt kromosomen i endygns-optimeringen bestod av fyra delar, en del vardera för värmekraftverk, kärnkraftverk, inköp och försäljning. Varje del bestod i sin tur av 72 bytes, d v s 72 stycken 8-bitars tal. Se figur 5.3 nedan. Avkodningen av kromosomen skedde så att första 8-bitars gruppen i värmekraftdelen beskrev körningsnivå timme 0, andra 8-bitars gruppen beskrev körningsnivå timme 1 o s v. Vattenkraften räknade vi precis som innan ut genom att beräkna skillnaden mellan tillförd och bortförd kraft vid belastningsnoden. På detta sätt täckte vi även nu upp återstoden av belastningen med hjälp av vattnet.



Figur 5.3 Kromosomstruturen för vår tredygnsoptimering

När implementeringen var klar, så började vi testa ut de olika parametrarna. Efter en stor mängd körningar kom vi fram till att bästa konvergens uppnåddes med uniform överkorsning, linjärt avtagande mutationssannolikhet, konstant straff och med vanlig binär kodning av talen. Parameterinställningarna för mutationssannolikheten blev 0.002, för migration 0.12 och för överkorsning 0.9. Vi prövade även olika storlekar på populationen och dess fördelning på olika delpopulationer. Det som gav det bästa resultatet, var en totalpopulation på 1800 individer med uppdelningen på 30 individer per delpopulation. Vi vill inte påstå att detta är den optimala populationsstorleken, eftersom våra datorers internminnen inte tillät oss att pröva större populationer.

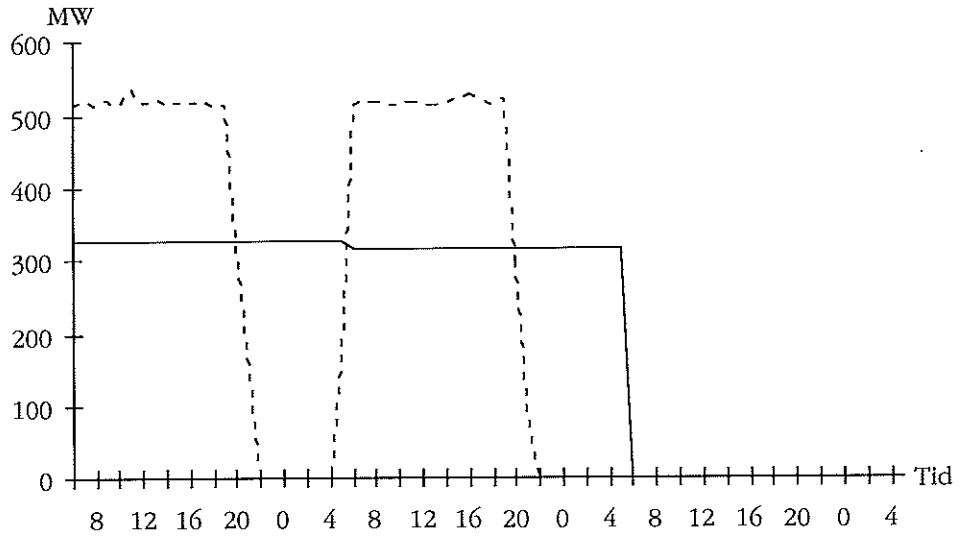
5.3 RESULTAT

I detta avsnitt presenterar vi en av de bästa lösningarna vi har fått fram på det problem vi har beskrivit tidigare i detta kapitel. Resultatet blev positivt i den bemärkelse att GA konvergerar mot optimum och visar att lösningen starkt liknar den manuella optimeringen man använder på Sydkraft idag. Tyvärr har det dock visat sig att GA tar oerhört lång tid på sig att lösa problem av denna storlek. Nedan visar vi utskriften från vårt GA-program.

```
NO GRAY - Lastkurva 3 - Genetiska parametrar:

Sannolikhet för överkorsning : 0.900000                    Uniform överkorsning
Sannolikhet för mutation     : 0.002000 Linjärt avtagande mutationssannolikhet
Sannolikhet för migration    : 0.120000                 Konstant straffunktion

Antal delpopulationer        :     60     Antal individer i varje     :     30
Antal generationer           :   4500     Kromosomlängd i bytes       :    288
Filnamn                      : rudolf1

Gen:    0  Max: 7.075282e+07  Avr: 6.728e+07  X:0.0e+00  M:0.0e+00  »:0.0e+00
Gen:  500  Max: 8.290670e+07  Avr: 8.189e+07  X:4.1e+05  M:4.0e+06  »:1.1e+05
Gen: 1000  Max: 8.324283e+07  Avr: 8.234e+07  X:8.1e+05  M:7.6e+06  »:2.2e+05
Gen: 1500  Max: 8.343563e+07  Avr: 8.258e+07  X:1.2e+06  M:1.1e+07  »:3.3e+05
Gen: 2000  Max: 8.353503e+07  Avr: 8.285e+07  X:1.6e+06  M:1.4e+07  »:4.4e+05
Gen: 2500  Max: 8.364506e+07  Avr: 8.308e+07  X:2.0e+06  M:1.6e+07  »:5.4e+05
Gen: 3000  Max: 8.377623e+07  Avr: 8.329e+07  X:2.4e+06  M:1.8e+07  »:6.5e+05
Gen: 3500  Max: 8.385504e+07  Avr: 8.347e+07  X:2.8e+06  M:2.0e+07  »:7.6e+05
Gen: 4000  Max: 8.394338e+07  Avr: 8.366e+07  X:3.2e+06  M:2.1e+07  »:8.7e+05
Gen: 4500  Max: 8.405299e+07  Avr: 8.385e+07  X:3.6e+06  M:2.2e+07  »:9.8e+05

Current time: 05:59:57 - 1993.08.11,  Executiontime: 36 h, 23 min, 17 sek
```

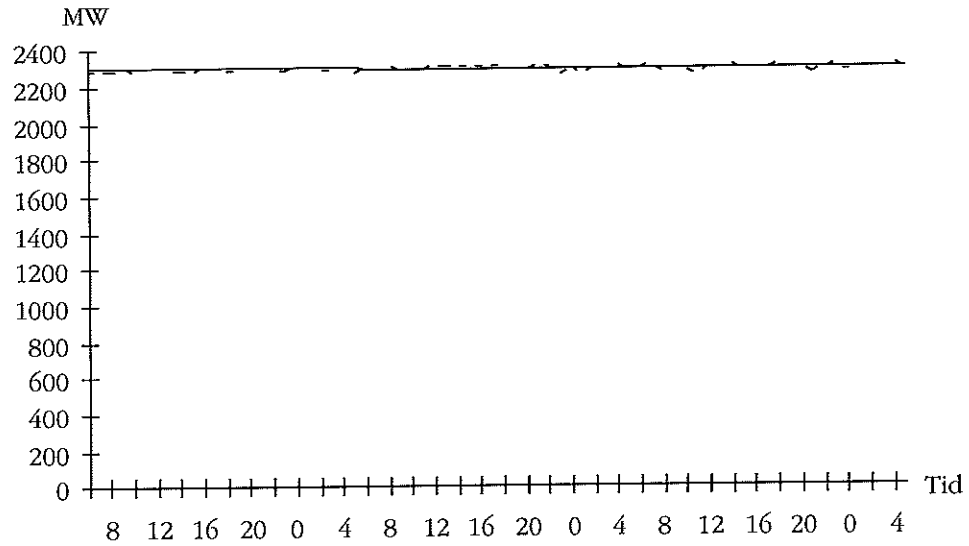
5.3.1 KÖRPLANER

Nedan presenterar vi de körplaner vi fått vid optimeringen av problemet. Det obearbetade resultatet som kommer direkt ut från GA-körningen presenteras som en streckad linje. I diagrammen har vi också lagt in medelvärdet över varje dygn som en heldragen linje. För vattenkraften har vi valt att beräkna medelvärden både för hög- och lågbelastningstid på dygnet samt tagit med dygnets min- och maxvärden.



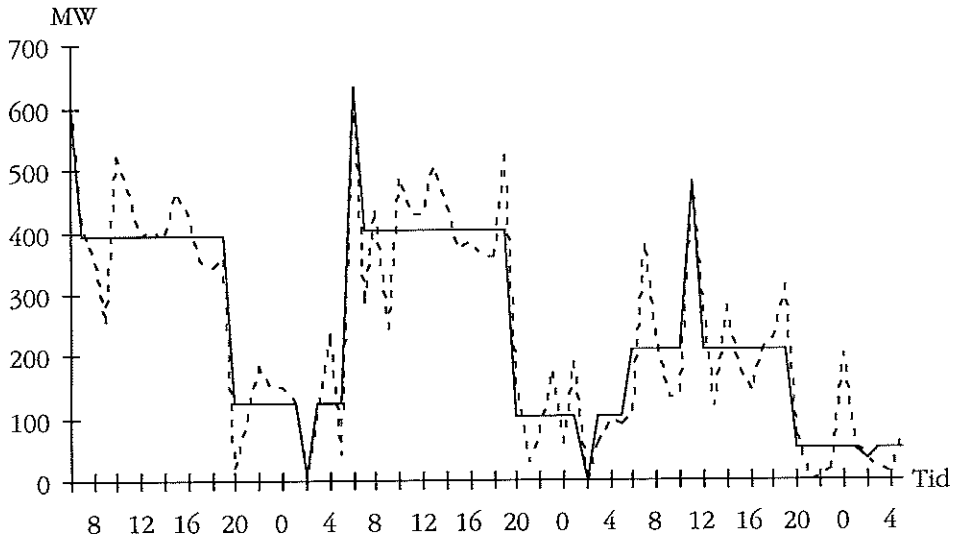
Figur 5.4 Värmekraftkörning enligt GA (streckad) och medelvärde per dygn (hel).

På värmekraftkörningen, i figur 5.4 ovan, ser man två uppstarter under tidsperioden, en på varje vardag under högbelastningstiden. Därutöver körs värmekraften inte. Det bör även noteras att GA inte överträder de körrestriktioner som existerar, d v s både uppstarter och nedgångar sker under kontrollerade former. Visst brus kan ses under körningen.



Figur 5.5 Kärnkraftkörning enligt GA (streckad) och medelvärde per dygn (hel).

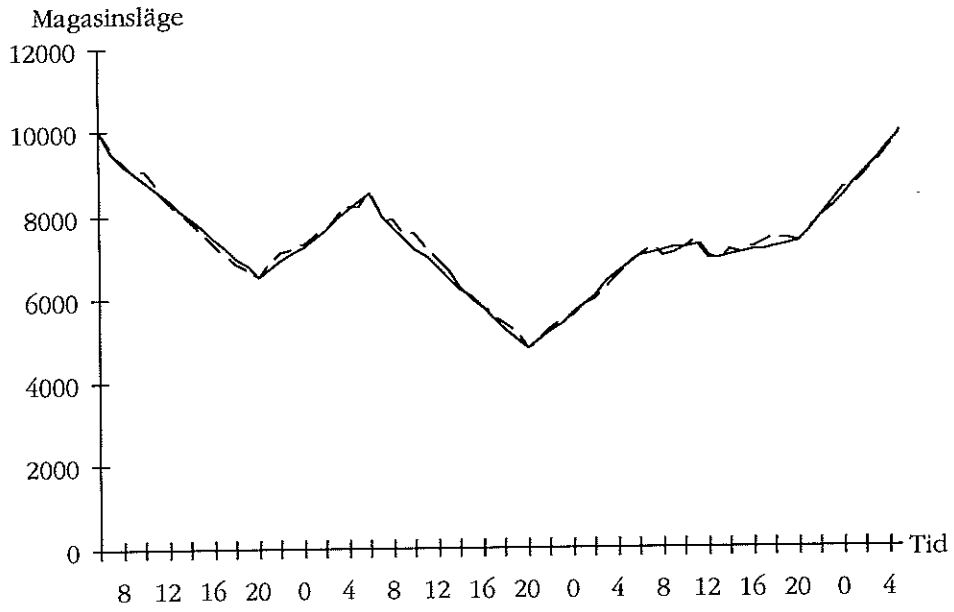
Det finns inte mycket att säga om kärnkraftkörningen, se figur 5.5 ovan. Kärnkraften körs med i princip full effekt under hela perioden. Visst störande brus kan dock ses.



Figur 5.6 Vattenkraftskörning enligt GA (streckad) och medelvärde (hel).

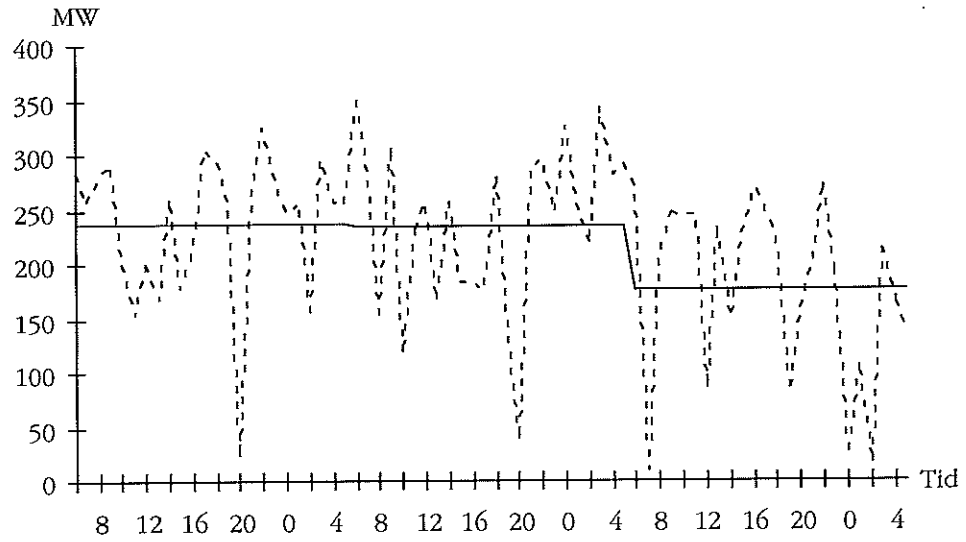
Vattenkraften, se figur 5.6 ovan, är kraftigt störd av brus. Bildar man medelvärdet över dygnens hög- och lågbelastningsperioder samt tar med dygnsminimum och dygnsmaximum framträder dock ett tydligt mönster. De två vardagarna kör nästan identiskt lika mycket vattenkraft. Värt att notera är även att man fördelar vattnet så att

större delen av vattenkraften produceras på vardagarna. Jämför gärna även den heldragna medelvärdeskurvan med belastningskurvans utseende, figur 5.2.



Figur 5.7 Magasinslägeenligt GA (streckad) och medelvärde (hel).

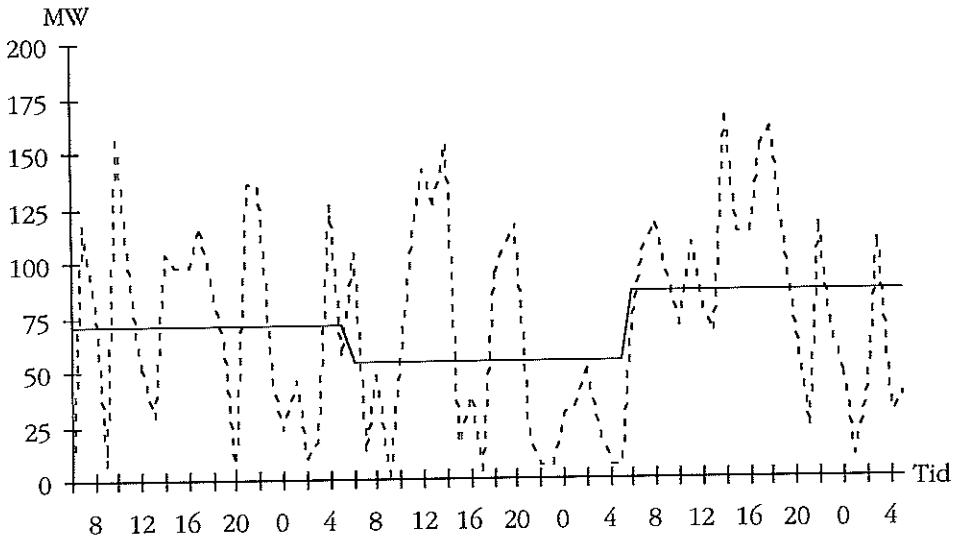
Figur 5.7 ovan visar hur magasinsläget förändras under de tre dagarna. Vi ser att man tappar vatten under dagtid de två vardagarna för att möta den höga belastningen. Under nätterna och under helgdagen sparar man sedan vatten så att man efter tre dagar är tillbaks i samma utgångsläge som när man startade.



Figur 5.8 Inköp enligt GA (streckad) och medelvärde per dygn (hel).

Inköpen, som visas i figur 5.8 ovan, är väldigt störda av brus. De totala inköpen under de två vardagarna överenstämmer dock. Dessutom är den genomsnittliga inköpsnivån

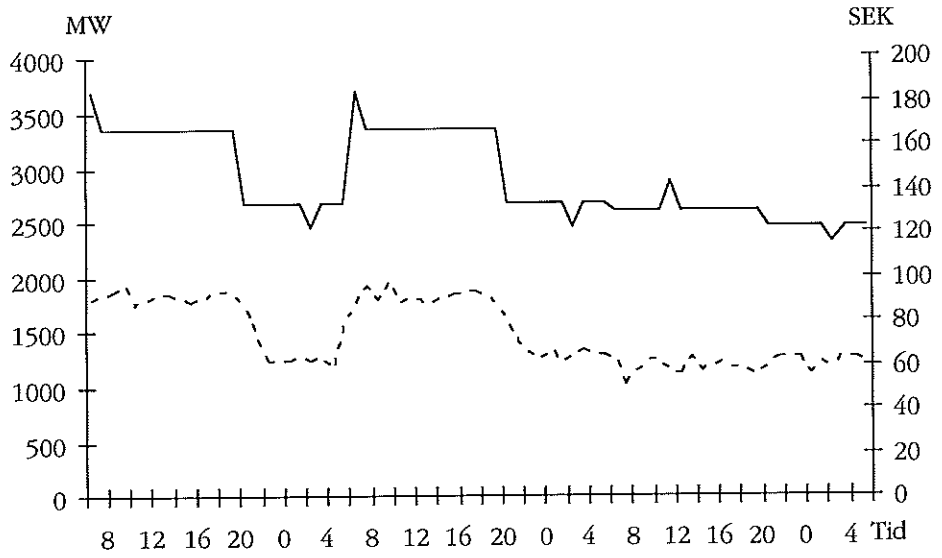
under helgdagen lägre än under vardagarna.



Figur 5.9 Försäljning enligt GA (streckad) och medelvärde per dygn (hel).

Även försäljningen, ovan i figur 5.9, är kraftigt störd av brus. Genomsnittsnivåerna över dygnen visas av de heldragna linjerna. Vi återkommer senare till analys av orsakerna till detta besynnerliga beteende.

Belastningen ser ut som figur 5.10 nedan visar. I figuren har vi även lagt in det genomsnittliga pris per producerad MWh. Notera hur tydligt belastningstopparna avspeglar sig i priskurvan.

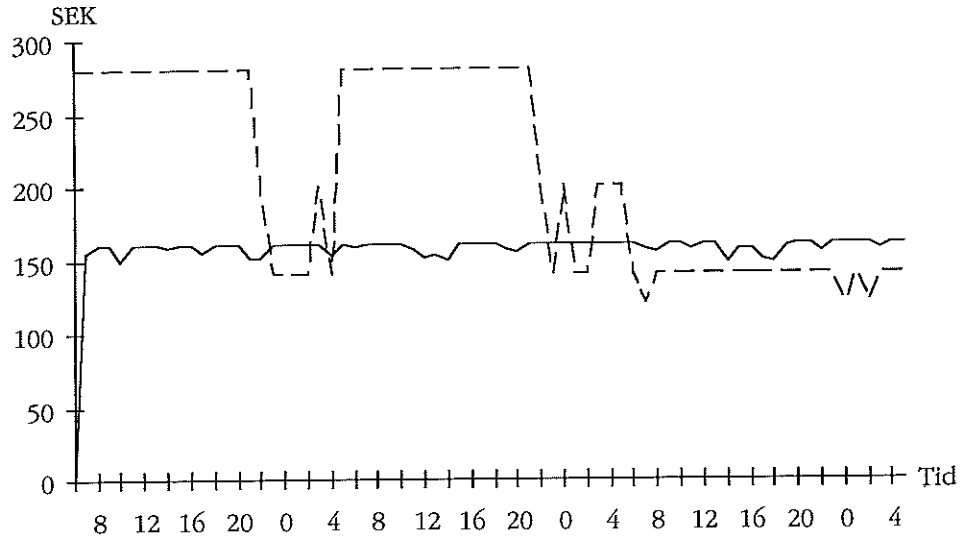


Figur 5.10 Belastningen (hel) och snittpris per producerad MWh (streckad).

5.3.2 FELKÖRNING

Tyvärr visar våra analyser att GA gör en del misstag. Dessa framträder speciellt tydligt hos försäljningsobjektet. Figur 5.11 nedan visar marginalkostnaden för att producera eller köpa den sista megawattimmen, samt aktuellt försäljningspris. Marginalkostnads-

kurvan ligger en mycket stor del av tiden över priskurvan. Detta innebär att all försäljning under dessa timmar går med förlust. Trots detta säljer GA en hel del kraft under dessa timmar. Vi har beräknat att förlusten p g a detta uppgår till ca 25 000 SEK under de tre dygnen eller 1.6 procent av den totala produktionskostnaden runt 16 miljoner kronor.



Figur 5.11 Marginalkostnad för sist producerad eller inköpt MWh (streckad) och genomsnittligt försäljningspris (hel).

Vi antar att anledningen till att GA säljer kraft med förlust är att förlusten är mycket liten i förhållande till den totala kostnaden. Vi misstänker att detta beror på att GA inte konvergerat tillräckligt. För att verifiera detta, studerar vi i nästa avsnitt hur GA konvergerar.

Förutom de brusiga inköps- och försäljningsplanerna anser vi att GA i huvudsak kör kraftverken rätt. Alla överträdelser av de begränsningar som finns i objekten som påverkar total kostnaden mest, d v s kärnkraft, värmekraft och vattenkraft, regleras bort av GA. Körplanerna överensstämmer även med de planer man får fram vid en marginalkostnadsanalys av den typ Sydkraft använder sig av idag.

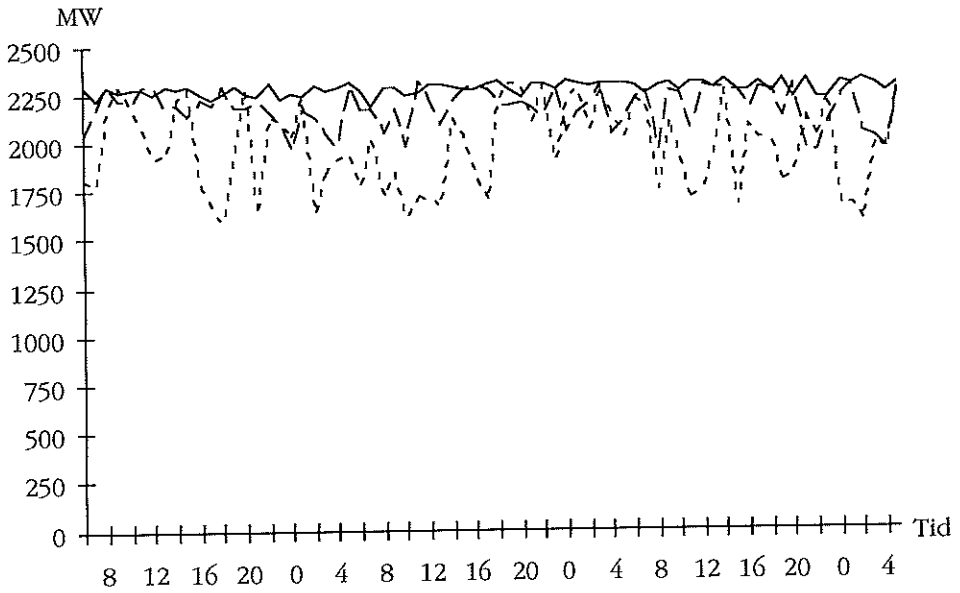
Marginalkostnadsanalysen bygger på att man producerar kraft med det kraftslag där marginalkostnaden är lägst tills man täcker belastningen. Detta innebär att man i första hand använder sig av vattenkraft och kärnkraft som har lägst marginalkostnad. I sista hand tar man till den dyra värmekraften. Vatten, som är lätt att reglera, används för att anpassa produktionen till en varierande belastning.

Sydkraft säljer kraft, när man har köpare som är villiga att betala mer än marginalkostnaden för kraften. I verkligheten, liksom i modellen, kan Sydkraft tjäna pengar genom att bara transportera kraft, d v s man köper kraft ifrån Norge och säljer den till Danmark. Detta kan i vår modell endast ske under förutsättningen att värmekraften inte kör. Endast då är marginalkostnaden för produktion och inköp lägre än försäljningspriset.

5.3.3 INSVÄNGNINGSFÖRLOPP

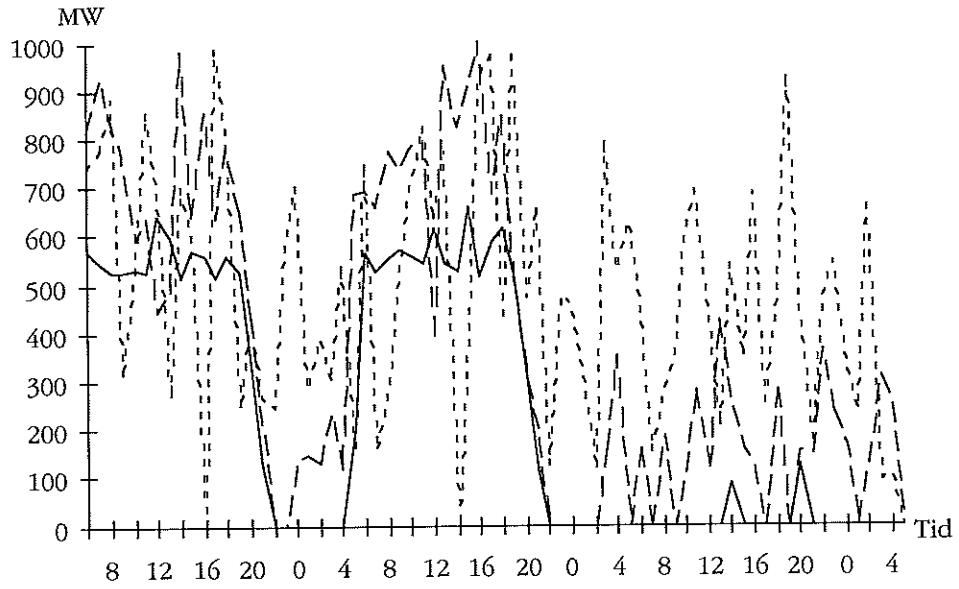
För att kunna förklara de missar GA har i sina förslag till körplaner, har vi tittat på körplanerna i olika konvergensstadier. Bildsviten nedan visar hur konvergensen går mot bättre och bättre värden. Bilderna visar den bästa individen i initialpopulationen

(punkter), bästa individen efter 100 generationer (streckad) samt bästa individen efter 1000 generationer (heldragen). Figur 5.12 nedan visar kärnkraftens insvängningsförlopp.



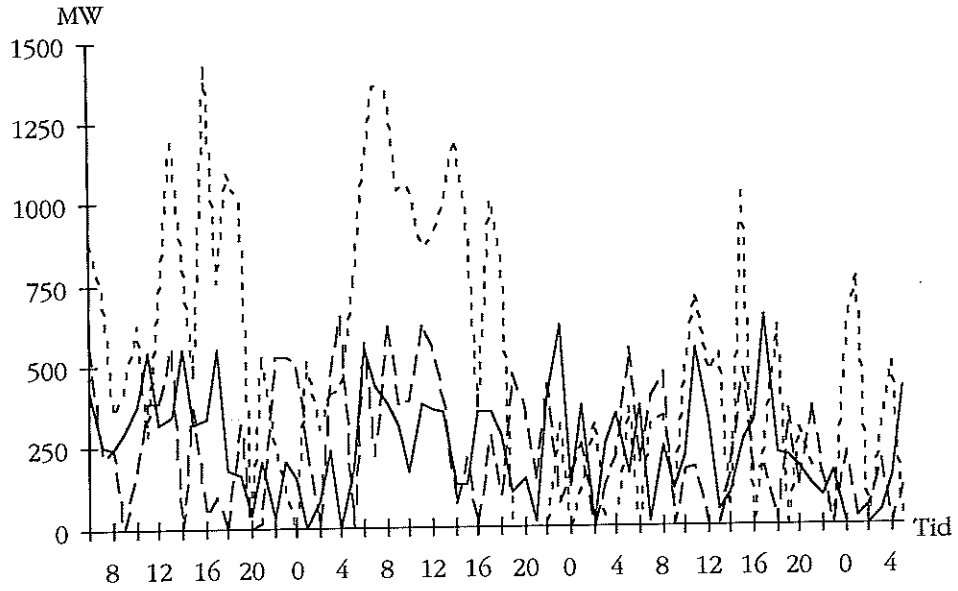
Figur 5.12 Körplan för kärnkraften efter 0 generationer (punktlinje), 100 generationer (streckad) samt 1000 generationer (heldragen).

Man ser tydligt i ovanstående figur hur kärnkraften svänger in sig mot kärnkraftverkets toppeffekt. Redan efter 100 generationer ser man tydligt i vilken riktning GA för körplanen. I figur 5.13 nedan visas hur värmekraftverket arbetar sig mot den optimala lösningen. Efter bara 1000 generationer ser man de två distinkta topparna i värmekraft-körplanen som motsvaras av vardagarnas högbelastningstimmar.



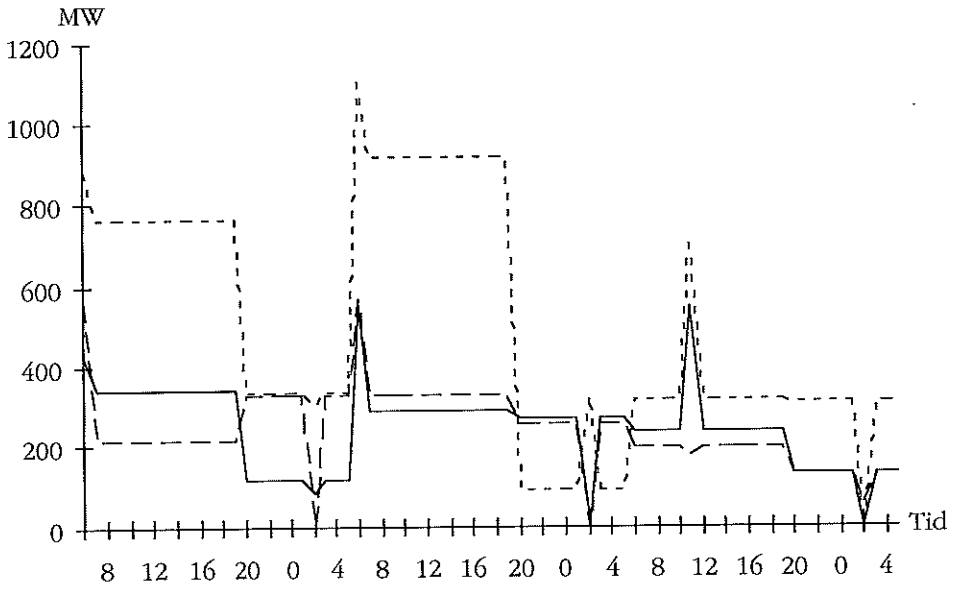
Figur 5.13 Körplan för värmekraften efter 0 generationer (punktlinje), 100 generationer (streckad) samt 1000 generationer (heldragen).

Vattenkraften visas i figur 5.14 nedan. Vid en första anblick ser man bara att körplanens snittvärde har sänkts mot de tillåtna tappningsgränserna. All övrig information är dränkt i brus.



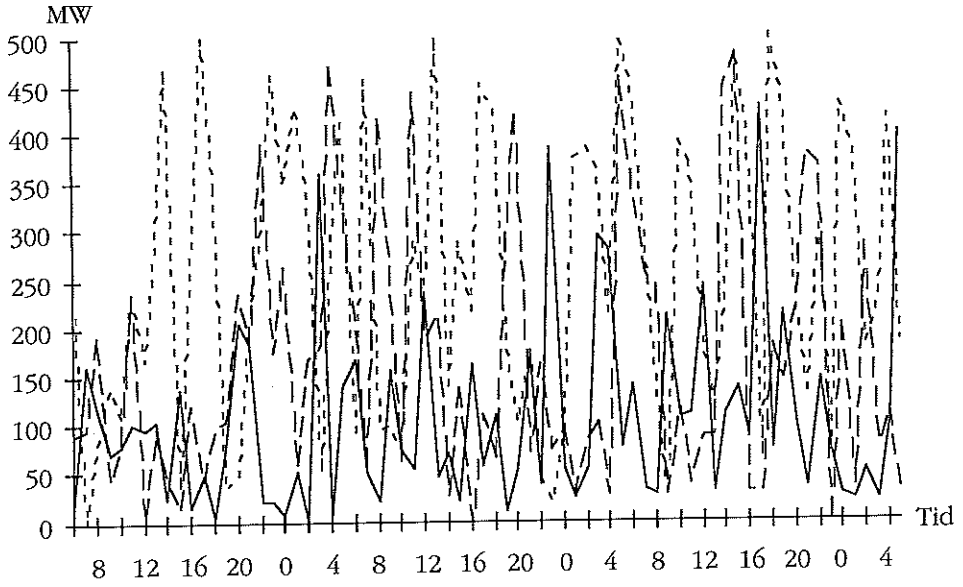
Figur 5.14 Körplan för vattenkraften efter 0 generationer (punktlinje), 100 generationer (streckad) samt 1000 generationer (heldragen).

Genom att medelvärdesbilda vattenkraftkörplanen över hög- och lågbelastningstimmarna samt ta ut dygnets min- och maxvärde, erhåller man kurvorna som visas i figur 5.15 nedan.



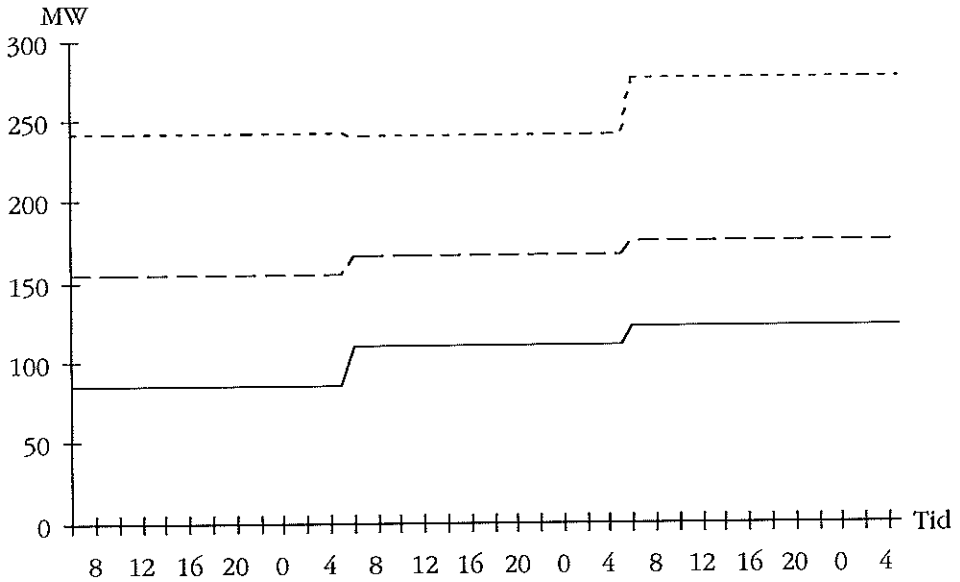
Figur 5.15 Medelvärdesbildad körplan för vattenkraften efter 0 generationer (punktlinje), 100 generationer (streckad) samt 1000 generationer (heldragen).

Nu framträder mönstret tydligare. Man ser tydligt hur medelnivån sänkts för att vattenkraftverket inte skall överskrida sin tappningsbegränsning. Vidare ser man hur vattenkraftskörningen anpassas mot belastningskurvan. Försäljningen som visas i figur 5.16 nedan är även den dränkt i brus.



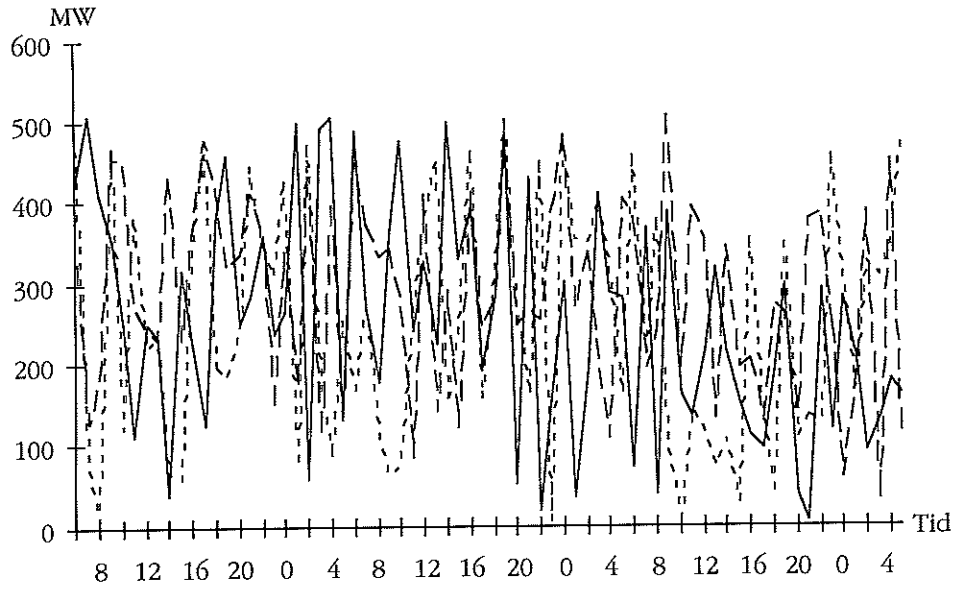
Figur 5.16 Körplan för försäljningen efter 0 generationer (punktlinje), 100 generationer (streckad) samt 1000 generationer (heldragen).

Återigen medelvärdesbildar vi körplanen för försäljningen, för att få fram trenderna. Denna gång gör vi det dock över de tre dygnen. Figur 5.17 nedan visar resultatet. Nu framträder ett tydligare mönster. Försäljningen snittnivå sänks, speciellt på de förlustbringande vardagarna. Helgdagens försäljning går med vinst och dess snittnivå ligger också över de övriga dagarnas.



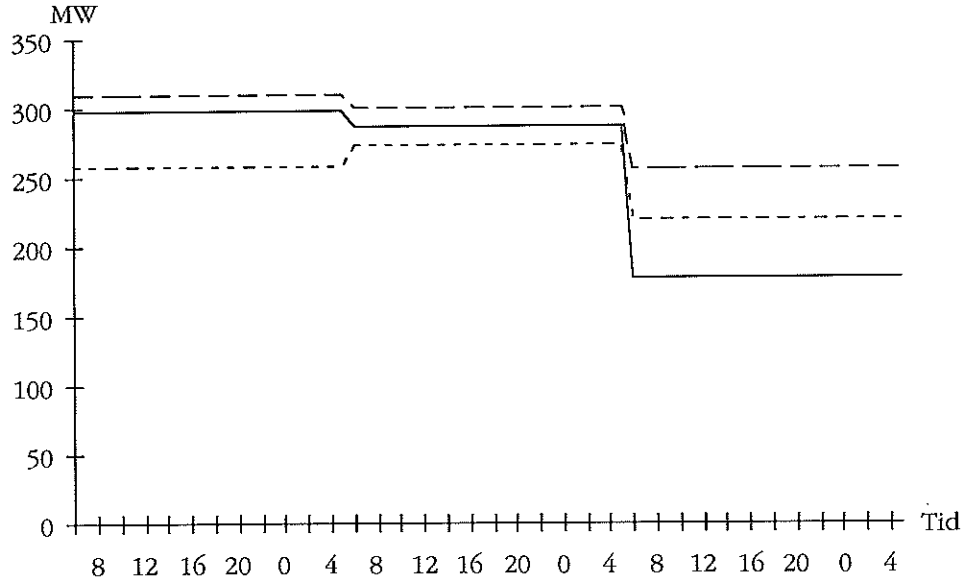
Figur 5.17 Medelvärdesbildad körplan för försäljningen efter 0 generationer (punktlinje), 100 generationer (streckad) samt 1000 generationer (heldragen).

Slutligen presenterar vi inköpen i figur 5.18 nedan. Det kraftiga bruset gör att man inte kan urskilja några väsentliga skillnader mellan körplanerna. Därför har vi än en gång medelvärdesbildat körplanen över dygnen i figur 5.19.



Figur 5.18 Körplan för inköpen efter 0 generationer (punktlinje), 100 generationer (streckad) samt 1000 generationer (heldragen).

Mönstret framträder nu tydligare. Genomsnittsnivåerna för inköpen under de två vardagarna ligger under hela optimeringen på en relativt konstant hög nivå. Inköpen under helgdagen däremot arbetar sig mot en lägre snittnivå.



Figur 5.19 Medelvärdesbildad körplan för inköpen efter 0 generationer (punktlinje), 100 generationer (streckad) samt 1000 generationer (heldragen).

Bildsviten i detta avsnitt har gett en större förståelse för hur GA svänger in sig mot den optimala lösningen. De objekt som har störst betydelse för totalkostnaden, svänger in sig väldigt snabbt. Detta illustreras i figurerna som visar kärnkraftens, värmekraftens samt vattenkraftens insvängningsförlopp. De övriga två objekten, försäljning och inköp, har en mindre inverkan på totalkostnaden och följaktligen svänger de in mycket långsamt. När vi avbröt optimeringen efter totalt 4500 generationer, var deras körplaner fortfarande dränkta i brus. Försäljningen innehöll även allvarliga felkörningar.

Om vi kopplar detta till vad vi känner till om algoritmkomplexiteten från kapitel 3 inser man lätt att mycket bättre lösningar än den framtagna, tar mycket längre tid att få fram. Ett rimligt antagande kan vara att vi efter 4500 generationer har kommit fram till en lösning som är 95 procent av optimum. Detta antagande ger att en 99 procentig lösning skulle ta över 200 timmar att få fram på en Macintosh Quadra 700.

5.4 SAMMANFATTNING

I detta kapitel har vi undersökt GA:s beteende på ett olinjärt kraftsystem. GA:s körplansresultat har vi utvärderat med hjälp av erfarenheter från kapitel 4. Vi har även låtit Sydkraftpersonal analysera resultatet. Deras och vår bedömningen är att GA-resultatet helt klart har närmat sig optimum, men att det fortfarande är en bra bit kvar.

GA-lösningen visar stora likheter med den manuella optimering Sydkraft använder idag och resultatet i kapitel 4. Detta är ett naturligtvis ett bra resultat men våra beräkningar visar att problemet tar väldigt långt tid att lösa fullständigt. Resultatet är nedslående i den bemärkelse att en fullskalemodell skulle ta orimligt långt tid att lösa. Trots detta anser vi att GA är en mycket kraftfull och intressant optimeringsmetod.

6. ERFARENHETER OCH SLUTSATSER

Det primära problemet, för Sydkraft och andra elproducenter är att tillgodose sina kunder med exakt så mycket elkraft de efterfrågar, så ekonomiskt som möjligt. Detta kan delas upp i två delproblem, dels att prognosticera efterfrågan så bra som möjligt och dels att utifrån denna bestämma hur produktionen skall ske så billigt som möjligt. Vårt examensarbete behandlar det sistnämnda av dessa två delproblem, d v s att ta fram planer för hur Sydkraft bör köra sina kraftverk och hur de bör köpa och sälja kraft för att maximera sin vinst.

Efter att ha arbetat med genetiska algoritmer [GA] på denna typ av problem har vi erhållit en intuitiv känsla för hur GA fungerar, vad man kan och vad man inte kan göra med GA. Våra resultat är blandat positiva och negativa. Det finns ingen tvekan om att GA är en mycket intressant och kraftfull optimeringsmetod. Vidare är metoden synnerligen robust och konvergerar mot värden nära optimum för de flesta problemtyper.

Även på våra problem konvergerar GA. Vid jämförelse med linjär programmering når GA en minimal totalkostnad som ligger mindre än 1.2 procent från optimum på ett linjärt kraftsystem. GA har dock fördelen att inte besväras av olinjäriteter, diskontinuerliga funktioner etc. Därför fungerar GA även tillfredsställande på vårt olinjära kraftsystem.

Våra problem är dock relativt svåra för GA att lösa. Det finns ett kraftigt beroende mellan kromosomens gener. Speciellt gäller detta upp- och nedregleringsbegränsningarna. Till exempel om vi har producerat 400 MW vid kärnkraftverket vid timme t, har vi begränsat nästa timmes "t+1":s produktionsnivå till intervallet [380,420] MW, timme "t+2":s produktionsnivå till intervallet [360,440] MW o s v. Samma problem finns även hos värmekraftverket. I verkliga livet förekommer regleringsbegränsningar hos alla produktionsanläggningar. En annan typ av beroende finns hos vattenkraften. Vi har bestämt oss för att tappa en viss mängd vatten under dygnet vilket medför att summan av alla körnivåerna skall vara ekvivalent med den mängd vatten som skall tappas. En enskild körnivå beror då på alla de övriga. Trots dessa svårigheter hittar GA alltid lösningar som inte överskrider de givna begränsningarna.

Det största problemet anser vi dock vara svårigheterna att skapa fristående byggblock inom kromosomen. För att man skall uppnå ett bra resultat med GA krävs att man kan bryta ner problemet i isolerade delproblem[12]. För att exemplifiera det antar vi att man vill att GA ska ta fram ett flygplan. En del av kromosomen innehåller vingar, en del motor och en del styrsystem. Alla delarna finns med i kromosomen men de kan optimeras var för sig och man får ett bra slutresultat. I vårt fall kan man inte betrakta de olika kraftverken som avskilda byggblock eftersom allting hänger samman. Vi kan inte optimera ett enskilt kraftverk utan att ta hänsyn till de övriga. Detta beroende mellan alla kromosomens delar leder till ett mycket svårlöst och långsamt konvergerande problem.

Ett annat svårighet är kostnadsfunktionens utseende nära optimum. Om man betraktar GA-lösningen mot LP-lösningen ser man stora likheter men också stora skillnader. Kostnadsfunktionen är mycket flack nära optimum och detta medför att man samtidigt som man är mycket nära i kronor mätt är mycket långt från den optimala lösningen. Detta leder till att konvergensen blir mycket långsam när man kommit i närheten av optimum eftersom lösningar runt omkring optimum skiljer sig väldigt lite från varandra.

Våra undersökningar av algoritmkomplexiteten är nedslående. GA har ingen möjlighet att optimera stora kraftsystem på rimlig tid. En total modell över Sydkraft skulle behöva innehålla minst 100 objekt och ha väsentligt fler begränsningar och därmed fler

---

[12]Goldberg David E, Making genetic algorithms fly, Artikel i Advanced Technology for Developers Volume 2 February 1993.

kromosomberoenden. En väldigt försiktig beräkning ger att en veckooptimering av en fullskalemodell skulle ta minst 10^3 år på en snabb dator.

Efter detta konstaterande har metoden mest ett akademiskt intresse för lösning av denna typ av problem. På mindre delproblem kan emellertid metoden användas med framgång. Vid alla GA-uppgifter har man problemet med parameterval. Eftersom teorin angående GA i bästa fall är sparsam saknas möjlighet att teoretiskt bestämma parametrarna. Det återstår då att prova olika parameteruppsättninga för att utröna vilken som är bäst. GA är visserligen en mycket robust optimeringsmetod som konvergerar även då parametrarna inte är speciellt väl valda, men konvergenshastigheten blir då starkt lidande och man når inte heller lika bra lösningar.

Slutligen kan vi bara konstatera att GA är en bra optimeringsmetod men fullskaleproblemets storlek och komplexitet resulterar i att GA, liksom många andra optimeringsmetoder, misslyckas med att lösa problemet inom en rimlig tid.