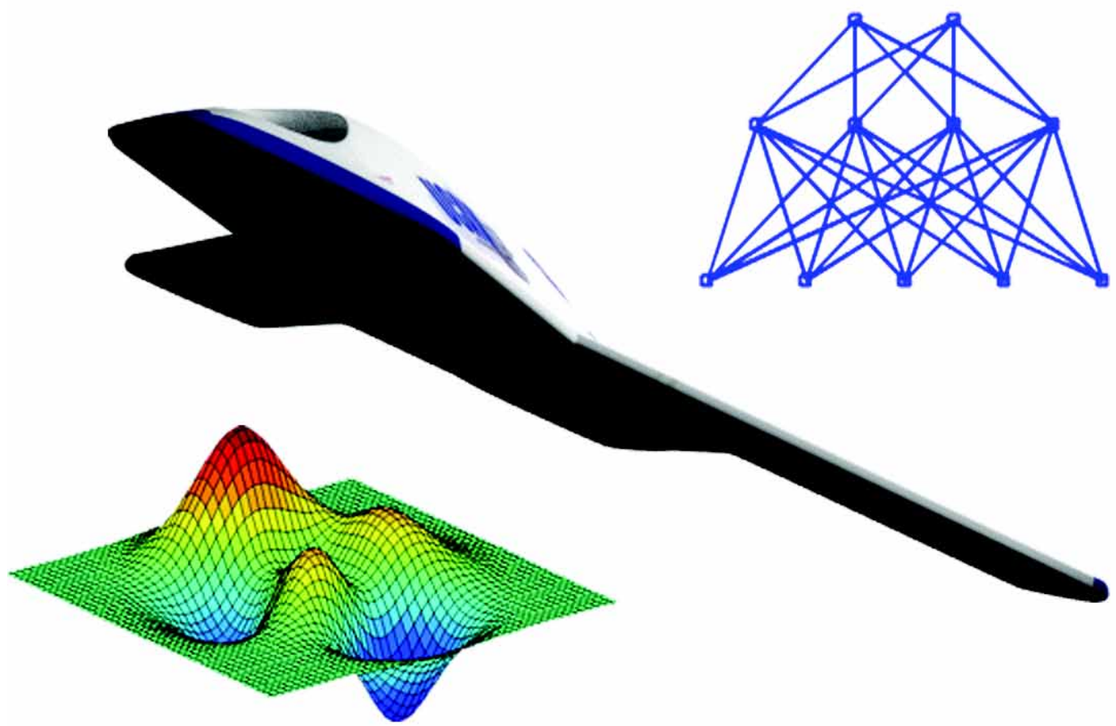
Att förstå Artificiell Intelligens genom Uppdragstaktik och Optimeringslära

PETTER ÖGREN



FOI
Totalförsvarets forskningsinstitut
Försvars- och säkerhetssystem
164 90 STOCKHOLM

Tel: 08-555 030 00
Fax: 08-555 031 00

www.foi.se

FOI-R--2262--SE Användarrapport
ISSN 1650-1942 April 2007

Försvars- och säkerhetssystem

Petter Ögren

Att förstå Artificiell Intelligens genom Uppdragstaktik och Optimeringslära

FOI-R--2262--SE Användarrapport **Försvars- och säkerhetssystem**
ISSN-1650-1942 April 2007

Utgivare FOI – Totalförsvarets forskningsinstitut Försvars- och säkerhetssystem 164 90 STOCKHOLM	**Rapportnummer, ISRN** FOI-R--2262--SE	**Klassificering** Användarrapport
	Forskningsområde Ledning och MSI	
	Månad år April 2007	**Projektnummer** E62156
	Delområde Ledning	
	Delområde 2	
Författare/redaktör Petter Ögren	**Projektledare** Petter Ögren	
	Godkänd av Helena Bergman	
	Uppdragsgivare/kundbeteckning FMV	
	Tekniskt och/eller vetenskapligt ansvarig Martin Hagström	
Rapportens titel Att förstå Artificiell Intelligens genom Uppdragstaktik och Optimeringslära		

Sammanfattning

Artificiell Intelligens (AI) är benämningen på en blandad uppsättning algoritmer som allt oftare används i obemannade farkoster och andra komplexa moderna vapensystem. Tanken på intelligenta maskiner gör att dessa algoritmer gärna omgärdas av en aura av mystik och Hollywood-visioner. Det senare står i skarp kontrast till användares och beställares behov att kunna förstå systemens möjligheter och begränsningar.
Syftet med denna rapport är att möjliggöra en konkret och avmystifierad bild av ett AI-system, med hjälp av optimeringslära och idéen bakom uppdragstaktik. Försvarsmaktens ledningsmetod, uppdragstaktik, har nämligen mycket gemensamt med grundtanken i teorin för automatiserat beslutsfattande, optimeringslära, och när vi ser AI-metoderna som optimeringsmetoder framträder både för- och nackdelar tydligt.
För att konkretisera sambanden ovan föreslår vi fem frågor som man kan ställa leverantören av ett AI-system. Tolkningar av möjliga svar, samt motiveringar till frågorna presenteras i rapportens olika delar. Med hjälp av svaren på frågorna får man snabbt en översiktlig och objektiv bild av både styrkor och svagheter hos det aktuella systemet.

Nyckelord AI, Artificiell Intelligens, Optimeringslära, Uppdragstaktik, Neurala Nätverk, Genetiska Algoritmer, Konvexitet, Komplexitet, UAV, UGV, Robotar, Autonomi	
Övriga bibliografiska uppgifter	**Språk** Svenska
ISSN ISSN-1650-1942	**Antal sidor:** 22 s.
Distribution enligt missiv	**Pris: Enligt prislista**

Issuing organisation FOI – Swedish Defence Research Agency Defence and Security, Systems and Technology SE-164 90 STOCKHOLM	**Report number, ISRN** FOI-R--2262--SE	**Report type** User report
	Research area code Mobility and Space Technology, incl Materials	
	Month year April 2007	**Project no.** E62156
	Sub area code Unmanned Vehicles	
	Sub area code 2	
Author/s (editor/s) Petter Ögren	**Project manager** Petter Ögren	
	Approved by Helena Bergman	
	Sponsoring agency Swedish Defence Materiel Administration	
	Scientifically and technically responsible Martin Hagström	

Report title
An Optimization Perspective on Artificial Intelligence

Abstract
Artificial Intelligence (AI) is the common label for a wide selection of algorithms that sometimes find their way into unmanned vehicles and modern weapon systems. Due to Hollywood movies and other parts of the popular science fiction culture, most people do not know what to expect form these algorithms. This fact is unsatisfactory, both from an operator and from a decision maker point of view.
The purpose of this report is to provide a tool that facilitates a clear and transparent view of the limitations and capabilities of AI algorithms.
This tool is a set of questions to be answered by the supplier of a military AI system. After stating the questions, the rest of the report is aimed at motivating them, and at giving interpretations of possible answers. All questions originate from the observation that AI and optimization theory have a lot in common, and viewing an AI-method as an optimization algorithm enables a clearer view of its advantages and drawbacks.

Keywords
AI, Artificial Intelligence, Optimization, Neural Networks, Genetic Programming, Convexity, Complexity, UAV, UGV, Autonomy

Further bibliographic information	**Language** Swedish
ISSN ISSN-1650-1942	**Pages** 22 p.
	Price According to price list

Innehåll

1 Introduktion

Obemannade vapensystem är ett allt vanligare inslag i militära operationer idag. Många av de system som används i fält är emellertid fortfarande *fjärrstyrda* i hög grad, men krav på minskad operatörsbelastning, ökad kostnadseffektivitet och bättre prestanda gör att metoder från autonomi- och AI-forskning allt oftare kommer till användning.

För en beställare kan det emellertid vara svårt att kravställa prestandan hos ett ”intelligent” system, och för en användare är det kanske ännu svårare att interagera med, och förutse beteendet hos, sådana. Svårigheterna i att hantera högt automatiserade system har till och med fått ett namn, *automation surprise*.

Syftet med denna rapport är att ge beställare och användare en möjlighet att på ganska kort tid, och med begränsade förkunskaper, översiktligt bedöma prestandan hos ett AI-system. Verktyget vi föreslår är fem korta frågor man kan ställa leverantören av systemet.

De fem frågorna återfinns i kapitel 2. Motivering till dem, och bakgrund till vilka svar som kan förväntas och hur de kan tolkas, finns i resten av rapporten.

I kapitel 3 kommer vi att dra paralleller mellan artificiell intelligens (AI) i militära system, ledningsmetoden *uppdragstaktik*, och teorin för automatiserat beslutsfattande, *optimeringslära*, samt argumentera för att det senare är ett sätt att länka samman de första två, som ger fördelar för både beställare och användare av AI-system.

För att tolka möjliga svar studerar vi i kapitel 4 optimeringsläran lite närmare, och ser exempel på flera stora problemklasser, av vilka några är extremt svårlösta, och andra överraskande lättlösta.

Kapitel 5 motiverar frågorna ytterligare, genom att ge exempel på AI-metoder och se hur deras för- och nackdelar kan förstås ur ett optimeringsperspektiv.

Slutligen återfinns en sammanfattning i kapitel 6.

2 Fem frågor rörande ett givet AI-system

Hur kan man då som lekman bedöma prestandan hos ett AI-system? Genom att ställa ett antal frågor till leverantören kan man få en tydlig bild av systemets intelligens. Om man dessutom känner till några få allmängiltiga teoretiska begränsningar för AI-system, se kapitel 4, kan man avgöra om det leverantören lovar är rimligt. Följande frågor bör en leverantör kunna svara på:

1. Formulera, i termer av uppdragstaktik, uppgiften AI-systemet löser!
2. Hur skiljer man en bra lösning från en dålig? (Målfunktion?)
3. Vilka krav ställs på en lösning? (Bivillkor?)
4. Garanteras att bästa möjliga lösning hittas? (Optimalitet?)
5. Om systemet dessutom innehåller ett lärande delsystem:
    a) Vilka parametrar ställs in av inlärningen?
    b) Hur skiljer man ett bra parameterval från ett dåligt? (Målfunktion? Denna borde ha en tydlig koppling till Målfunktionen för hela systemet)

Fråga 1 bygger på att uppgiften ett militärt AI-system löser rimligtvis måste vara minst lika tydlig som uppgiften som ställs till en människa, se avsnitt 3.2.

Fråga 2 och 3 motiveras i avsnitt 3.3, där sambanden mellan uppdragstaktik och optimeringslära diskuteras.

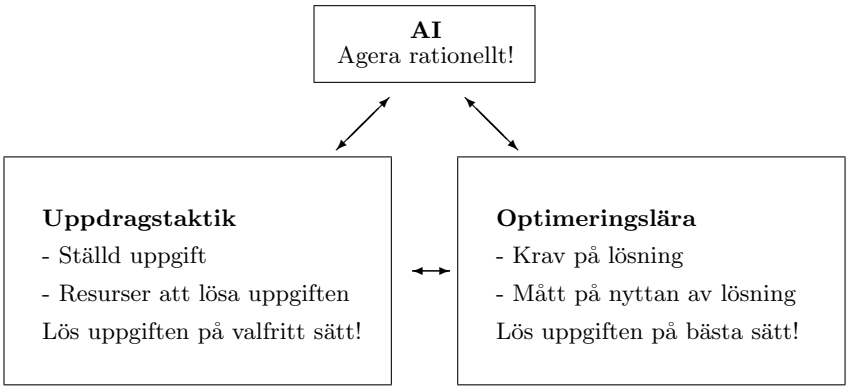
Fråga 4 motiveras av att det inom optimeringsläran finns ett antal stora problemklasser av olika svårighetsgrad, och svaret på frågan bestäms i stor grad av i vilken klass problemet ligger. De viktigaste problemklasserna beskrivs i avsnitt 4.4 nedan.

De sista frågorna, i punkt 5, tydliggör för användarna vilken typ av inlärning det handlar om. Alla lärande system inom AI kan nämligen ses som olika sätt att ställa in parametrarna i en ickelinjär funktion. Detta diskuteras i avsnitt 5.4.

Vi börjar enligt ovan med att motivera frågorna 1 till 3 i nästa kapitel.

3 AI, Uppdragstaktik och Optimeringslära

I detta kapitel kommer vi att beskriva sambanden mellan AI i militära system, uppdragstaktik, och optimeringslära. I figur 3.1 illustreras några kärnpunkter inom de olika områdena.



Figur 3.1: I denna rapport skall vi beskriva och dra paralleller mellan AI, Uppdragstaktik och Optimeringslära. AI handlar om system som agerar rationellt och uppdragstaktik lägger fokus på att fritt lösa ställd uppgift med givna resurser. Inom optimeringläran anges både krav på lösning och mått på nyttan av densamma.

3.1 AI i militära system

För att undvika begreppsförvirring börjar vi med att definiera vad vi menar med artificiell intelligens, AI, i detta sammanhang. Området AI är ganska brett, och i forskarkretsar finns ett antal olika föreslagna definitioner på exakt vad området innefattar. I boken *Artificial Intelligence: A Modern Approach*, [5], delas dessa definitioner upp i fyra huvudkategorier, se tabell 3.1.

System som tänker som människor.	System som tänker rationellt.
System som agerar som människor.	System som agerar rationellt.

Tabell 3.1: Definitioner av AI

Eftersom man i militära system är mer intresserad av systemprestanda än filosofiska spörsmål är rimligen den fjärde varianten den mest relevanta: *System som agerar rationellt*. Vad är då ett sådant system? Återigen kan vi kringgå de filosofiska fallgroparna genom att notera att vi helt enkelt vill ha ett system som löser uppgiften. Helst skulle man vilja ha ett system som löser uppgiften

på bästa möjliga sätt, med givna resurser och inom givna tidsramar. Eftersom detta börjar låta som en militär order gör vi nu en kort avstickare inom området militär ledning och uppdragstaktik.

3.2 Uppdragstaktik

Det svenska militära försvarets ledningsmetod kallas *uppdragstaktik*, och sammanfattas i skriften Grundsyn Ledning, [1], på följande vis. Uppdragstaktik innebär att en chef

- ställer uppgift,
- tilldelar resurser, samt
- ger den som skall lösa uppgiften största möjliga frihet att välja *hur* uppgiften skall lösas.

Eftersom denna ledningsmetod tillämpas framgångsrikt i hela den militära ledningsstrukturen, vore det idealiskt att även kunna ge uppdrag till AI-system på samma sätt, dock inom systemets mycket begränsade tillämpningsområde.

Oavsett om systemet verkligen kan ta emot en order i termer av uppdragstaktik, måste ordern i sig vara möjligt att formulera på detta sätt. För att tydliggöra vad det är det ”intelligenta” systemet skall göra föreslår vi alltså att man formulerar uppgiften med hjälp av uppdragstaktik, se fråga 1 i kapitel 2. Om ordern inte kan formuleras på detta sätt är uppgiften för luddig, vilket omöjliggör en utvärdering av AI-systemets prestanda.

En maskin måste förstås få sin uppgift och resurser tilldelad på ett väldigt strukturerat sätt, men idéen att tilldela en maskin en uppgift och sedan ge den största möjliga frihet att välja lösningsalternativ är i själva verket inte ny, den har funnits länge inom teorin för automatiserat beslutsfattande, optimeringslära.

3.3 Optimeringslära

Optimeringslära är namnet på teorin för automatiserat beslutsfattande. Optimeringsmetoder dök första gången upp under andra världskriget för att analysera och förbättra amerikanska försvarets logistikkedjor. Denna gren av optimeringslära kallas fortfarande *operations research*, men sedan dess används optimeringsmetoder också inom t.ex.

- Flygplansdesign, för att få bästa prestanda under givna hållfasthetskrav
- Strålbehandling av cancer, för att minimera biverkningarna samtidigt som tumören slås ut
- Ekonomi, att maximera den förväntade vinsten i aktieportföljer under givna krav på risker.
- Planering av raketbanor, för att kunna skicka upp så stora satelliter som möjligt i given omloppsbana
- Kabinpersonalsplanering, med så få anställda som möjligt genomföra alla utlovade flygrutter, utan att bryta mot regler om piloters sömn etc.

Vad som är gemensamt för alla exempel ovan, är uppdelningen i två delar, det som skall vara så bra som möjligt, *målfunktionen*, och det som måste uppfyllas, *bivillkoren*. Eller annorlunda uttryckt:

- Målfunktion: Mått på hur bra en lösning är.
- Bivillkor: Krav som ställs på lösningen.

Låt oss nu återigen titta på exemplen ovan i lite mer detalj.

Problem	Målfunktion	Bivillkor
Flygplansdesign	Maximera räckvidden	Givet krav på nyttolast och säkerhet etc.
Strålbehandling	Minimera skadan i frisk vävnad	Givet viss stråldos i tumören
Ekonomi	Maximera den förväntade vinsten	Givet gräns för risktagande
Raketbanor	Maximera lastkapaciteten	Givet att hela lasten når rätt omloppsbana
Kabinpersonalsplanering	Minimera antalet piloter och flygvärdinnor	Givet att alla flygningar kan genomföras

Tabell 3.2: Exempel på problem, målfunktioner och bivillkor.

Jämför vi nu optimeringslära med uppdragstaktik ser vi att den ställda *uppgiften* motsvaras av både *målfunktion* och *bivillkor*, de *tilldelade resurserna* motsvaras av ytterligare *bivillkor*, samt att *största möjliga frihet*, svarar mot att inga ytterligare inskränkningar görs. Givet målfunktion och bivillkor skall algoritmen hitta en lösning som uppfyller bivillkoren med ett så stort (maximering), eller litet (minimering), värde på målfunktionen som möjligt.

Innan man matar in ett optimeringsproblem i datorn måste man dock specificera målfunktion och bivillkor exakt i form av programkod eller matematiska formler. Samtidigt som man gör detta avgör man vilka typer av lösningar datorn skall leta bland. Detta svarar mot att tilldela resurser.

Vår slutsats blir alltså följande: Om vi kan formulera AI-problemet som ett optimeringproblem blir det tydligt

- Vad vi kräver av lösningen: bivillkor
- Vad vi menar med att en lösning är bättre än en annan: målfunktion

Systemförståelse och systemtilltro blir då bättre än om systemets operatör bara vet att algoritmen är “intelligent”. Detta motiverar frågorna 2 och 3 i kapitel 2.

3.4 Likheter mellan AI och Optimeringslära

Ovan argumenterar vi för att användning av optimeringsformuleringar ger avsevärda fördelar för AI systemets operatör. Gör då detta att vi måste använda helt andra lösningsmetoder? Nej, i detta avsnitt skall vi först notera att ett antal militära AI-problem går alldeles utmärkt att formulera som optimeringsproblem, sedan skall vi räkna upp ett antal AI-metoder som i själva verket är optimeringsmetoder.

3.4.1 AI-problem som Optimeringsproblem

- Attackuppdrag med UCAV (unmanned combat air vehicle) Målfunktion: Maximera överlevnadssannolikhet. Bivillkor: Givet att lasten fälls över målet på utsatt tid, no-fly zoner inte överträds, bränslemarginal uppfylls etc.

- Spaningsuppdrag med UAV. Målfunktion: Maximera kvaliteten på sensordata. Bivillkor: Givet att tillåtna bränslemarginaler och risknivåer inte överskrids.
- Med en UGV samtidigt kartera och navigera i en okänd omgivning. Målfunktion: Minimera felen i den skapade kartan och osäkerheten i skattning av egen position. Bivillkor: Inga.
- Med UAV och UGV jaga en rörlig inkräktare (s.k. Pursuit Evasion Game) Målfunktion: Minimera förväntad tid till infångande. Bivillkor: Inga

Vi har nu sett att ett antal AI-problem går utmärkt att precisera som optimeringsproblem. Hur är det då med lösningsmetoderna?

3.4.2 AI-metoder som Optimeringsmetoder

När vi tittar närmare på ett antal AI-metoder ser vi att metoden i själva verket löser ett optimeringsproblem. Exempel på detta är t.ex. att när man tränar ett Neuralt nätverk löser man ett icke-linjärt optimeringsproblem med hjälp av *steepest decent*-metoden, se avsnitt 5.4, och när man löser ett problem med hjälp av Genetiska algoritmer, löser man återigen ett optimeringsproblem, denna gång med en metod som saknar prestandagarantier, se avsnitt 5.1.

En fördel med att se AI problem ur ett optimeringsperspektiv är att man även som lekman med hjälp av några få resultat ur optimeringsläran kan få en bild av systemets prestanda och begränsningar. Detta beskrivs närmare i nästa kapitel.

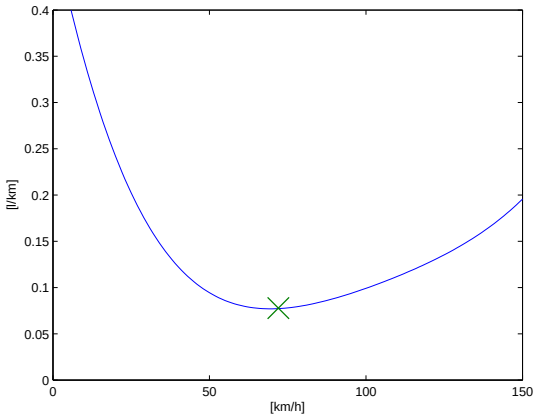
4 Optimeringslära: en kort översikt

I det här kapitlet skall vi se exempel på två olika typer av optimeringsproblem, *diskreta* och *kontinuerliga*. I diskreta problem skall man välja lösningen ur ett fixt antal alternativ, t.ex. ta norra eller södra vägen, och i kontinuerliga problem finns ett helt spektrum av lösningar att välja bland, t.ex. vilken hastighet man skall röra sig med.

Vi skall också se exempel på lösningsmetoder för de två fallen och lära oss hur man ser skillnaden mellan svårlösta och lättlösta problem av båda sorterna. Det sistnämnda kommer att motivera fråga 4 i kapitel 2. För att illustrera de olika typerna av optimeringsproblem betraktar vi följande exempel.

4.1 Kontinuerlig och diskret optimering för bilfärd från Uppsala till Norrköping

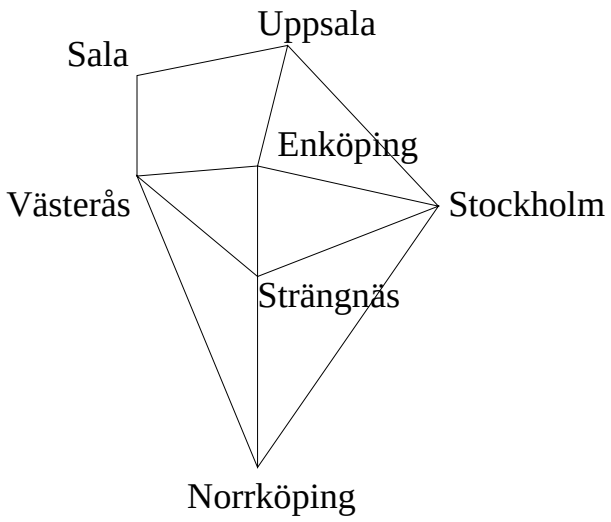
Antag nu att vi vill minimera bränslekostnaden för en bilresa mellan Uppsala och Norrköping. Bränsleförbrukningen för bilen finns uppritad i figur 4.1, där vi kan avläsa att just denna bil drar minst bensin vid den blygsamma hastigheten 73 km/h. I detta fall har vi ett oändligt antal möjliga val, alla hastigheter mellan 0 och bilens maxhastighet, sådana optimeringsproblem kallas *kontinuerliga*.



Figur 4.1: Bränsleförbrukningen för bilen i exemplet.

Att spara bensin handlar emellertid inte bara om att köra i rätt hastighet, utan också om att välja rätt väg. En illustration över några alternativa vägar mellan Uppsala och Norrköping finns i figur 4.2. Efter att ha betraktat kartan ser vi att vägen via Enköping förmodligen är den kortaste, men att den via Stockholm (E4:an) kanske är snabbare. När man på detta sätt väljer mellan ett fixt antal alternativ kallas optimeringsproblemet *diskret*. Detta är förstås

ett exempel på de vägvalsproblem som löses i kommersiella GPS-navigatorer för bilar.



Figur 4.2: Alternativa vägval mellan Uppsala och Norrköping

4.2 Varianter av optimeringsproblemen ovan

För att illustrera varför det är viktigt att tydligt ange både målfunktion och bivillkor skall vi räkna upp på ett antal varianter av väg- och hastighetsvalsproblemet ovan.

1. Målfunktion: Minimera bränsle
   Bivillkor: (inget)

2. Målfunktion: Minimera bränsle
   Bivillkor: kör varken för snabbt eller för långsamt,
   utan håll precis vägskyltens hastighet.

3. Målfunktion: Minimera tid
   Bivillkor: bilens maxhastighet

4. Målfunktion: Minimera tid
   Bivillkor: max 20 liter bensin och max 20km/h för fort.

I punkt 1 beskrivs orginalproblemet, där lösningen innebär att vi kör i 73 km/h längs den kortaste vägen. Att köra i 73 km/h på motorväg är kanske inte att rekommendera, så i punkt 2 lägger vi till kravet att hålla den givna hastigheten. Då blir den optimala vägen kanske delvis en annan, med fler 70-vägar. Ett helt annat problem får vi om vi istället låter målfunktionen vara tid, som i punkt 3. Då blir lösningen förstås att köra den kortaste vägen med gasen i botten hela tiden. Ytterligare en variant finns i punkt 4, där skall vi köra fort, men måste klara oss på de 20 liter bensin som finns i tanken. Kör vi för fort tar bränslet slut. Dessutom vill vi inte överstiga hastighetsbegränsningarna alltför mycket, för att inte riskera alltför höga straff.

Som synes kan man formulera en lång rad olika optimeringsproblem för att välja väg och hastighet från Uppsala till Norrköping. Vilket av dessa problem vi väljer beror på vad det vi är ute efter. Har vi ont om tid eller bensin (pengar)?

Detta är ett starkt argument för att formulera AI-problem som optimeringsproblem, se fråga 2 och 3 i kapitel 2. Ett ”intelligent” val kan vara vad som helst, men ett optimalt val är just optimalt, *med avseende på* valda bivillkor och vald målfunktion. Uppgiften är tydligt formulerad, samtidigt som friheten att välja lösning inte är onödigt inskränkt.

4.3 Optimeringsalgoritmer

Vi skall nu översiktligt beskriva några algoritmer för att lösa optimeringsproblem. Vi börjar med att betrakta *heuristiker* en klass av algoritmer som är lätta att förstå och använda, men som ofta inte hittar den optimala lösningen. Sedan tittar vi på algoritmer som garanterat hittar den optimala lösningen, först för kontinuerliga och sedan diskreta problem.

4.3.1 Snabbt och hyfsat men inte optimalt: Heuristiker

Eftersom en dator kan testa ett stort antal lösningar på kort tid kan man hitta hyfsade lösningar till en ganska stor klass av problem genom att pröva sig fram, och fokusera letandet i närheten av de hittills bästa hittade lösningarna. I princip följer de flesta heuristiker följande schema:

1. Starta med en eller flera slumpvis valda lösningar
2. Utvärdera nya lösningar i *närheten* av de gamla
3. Släng de sämsta av de nya och gamla lösningarna
4. Om den senaste förbättringen är för liten, eller om ett visst antal iterationer genomförts så avsluta sökningen och använd den hittills bästa lösningen. Annars, fortsätt från punkt 2

Viktiga skillnader mellan olika heuristiker är om man utvärderar en eller flera lösningar parallellt, hur man hittar nya lösningar i närheten av gamla, och vilka lösningar man slänger. Exempel på heuristiker är *Genetiska algoritmer*, *Simulated Annealing* och *Tabu-search*, vilka behandlas i mer detalj i avsnitt 5.1, 5.2 och 5.3. Styrkan med heuristikerna är att man inte behöver veta något om strukturen hos optimeringsproblemet. Detta gör att metoderna är lätta att tillämpa på en bred klass problem. Priset man betalar för detta är att man inte kan veta om metoden hittar optimum, förutom då beräkningstiden är oändlig. Utan insikt i problemet kan man inte veta om man hittat en optimallösning, och inte heller veta vilken som är den bästa sökriktningen.

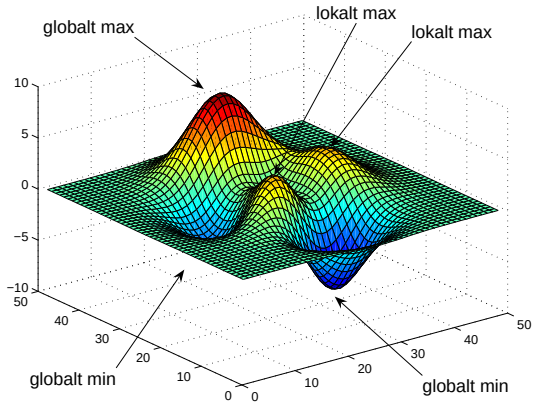
Innan vi går in på detaljer kring optimeringsalgoritmer som använder mer probleminformation för de olika kategorierna kontinuerliga och diskreta problem, noterar vi att heuristikerna ovan går att tillämpa på båda, bara man anpassar sättet att hitta nya lösningar *nära* de bästa av de gamla.

4.3.2 Algoritmer för kontinuerliga optimeringsproblem

Om vi återigen tittar på figur 4.1 ser vi att den optimala hastigheten kännetecknas av att den ligger längst ned i kurvans “grop”. En möjlig algoritm för att hitta sådana gropar, i en eller flera dimensioner, blir då att låta vår kandidatpunkt ”glida” längs kurvan tills den hamnar i en grop. Detta fungerar utmärkt i godtyckligt antal dimensioner, och kallas *steepest decent*-optimering (eller gradient-sökning, eftersom glidriktningen fås genom att ta gradienten av målfunktionen). Det finns ett antal förbättringar av metoden ovan, t.ex. Newtons metod, SQP (Sequential Quadratic Programming), och inrepunktsmetoder, se [2]. Gemensamt för dessa är att de till skillnad från Heuristikerna

använder information om lutningen (derivatan) och ibland även krökningen (andraderivatan) hos målfunktion och bivillkor. Detta möjliggör två saker, dels att man hittar en sökriktning som man vet ger en förbättring, och dels att man kan testa om man verkligen har hittat en lokalt optimal lösning.

Om det är fler än en variabel vi söker värdet på kallas problemet *flerdimensionellt*. Om man t.ex. tänker sig att man plottar vår exempelbils bensinförbrukning som funktion av både hastighet och yttertemperatur (bilen har luftkonditionering) får vi en tvådimensionell yta där vi kan söka efter minimum och maximum. I figur 4.3 ser vi exempel på en sådan yta, som dock inte visar bensinförbrukning.



Figur 4.3: I ett tvådimensionellt kontinuerligt optimeringsproblem bildar målfunktionen en yta. Målfunktioner kan ha både globala och lokala maximum och minimum.

Om vi använder en *steepest decent*-algoritm för att hitta maximat i figur 4.3 stöter vi på följande problem: vi kan hamna i vilken av de tre topparna som helst, beroende på var vi startar sökningen. När väl vår sökpunkt nått en topp finns ju ingen riktning att glida uppåt i, och sökningen stannar. I figuren ser vi att den bortre toppen är högst, ett så kallat globalt maximum, medan de två andra topparna är lägre, men samtidigt högre än sina respektive omedelbara omgivningar, de kallas därför lokala maximum.

Det finns effektiva algoritmer som hittar lokala maxima, men trots att det i en figur är väldigt lätt att se vilket som är det globala maximat, är det för en optimeringsalgoritm väldigt svårt att veta om det i en helt annan del av lösningsrummet existerar en bättre lösning än det lokala maximum man funnit. Testet som nämndes ovan visar bara om lösningen är lokalt optimal.

Ansatser till att lösa detta problem är Heuristiker som t.ex. *Simulated Annealing* eller *Tabu Search*, vilka behandlas i avsnitt 5.2 respektive 5.3. Som nämnts ovan kan man emellertid bara hoppas att lösningen de hittar är optimal.

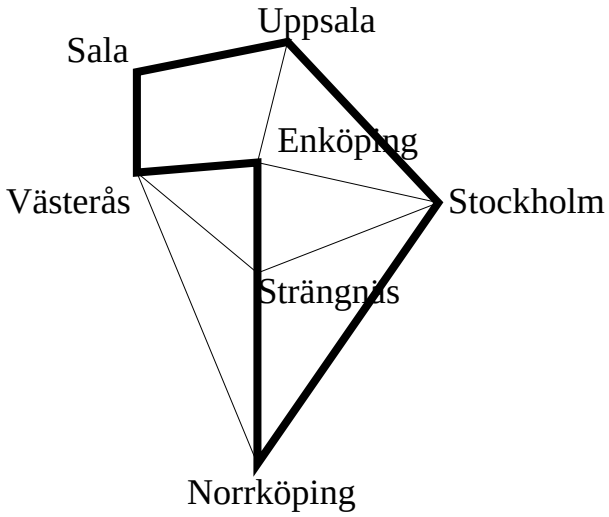
Problemet i figur 4.1 kallas konvext, och denna klass av problem är lätta att lösa med rätt metoder, även om de skulle ha 1000 variabler och hundra bivillkor. Metoderna man använder är de varianter av *steepest decent*-algoritmen som nämnts ovan.

Problem med många lokala extrempunkter, som det i figur 4.3, kallas icke-konvexa, och kan vara extremt svåra att lösa. Mer om detta kommer i avsnitt 4.4 nedan. Vi skall nu betrakta den andra typen av problem, de *diskreta*.

4.3.3 Diskreta optimeringsproblem

Diskreta optimeringsproblem handlar om att välja mellan ett fixt antal olika alternativ. Vägvalsproblemet vi såg i figur 4.2 var en förenkling av de större problem som dyker upp när man har alla vägar i en riktig karta att välja mellan. Dessa problem visar sig emellertid vara *väldigt lätta* att lösa, även med tusentals alternativa vägar, vilket är den teoretiska grundval som kommersiella GPS-navigatorer vilar på.

Ett besläktat, men märkligt nog *mycket svårare* problem är det s.k. handelsresandeproblemet (traveling salesman problem, TSP). Det handlar om hur en handelsresande skall resa för att besöka alla städer i ett givet område. I vår lilla exempelkarta skulle det kunna se ut som i figur 4.4.



Figur 4.4: En rutt som besöker alla städer i kartan.

Detta är också ett vägvalsproblem, men det tar märkligt nog avsevärt mycket längre tid att lösa. Därför kommer vi inte heller att se några GPS-navigatorer som löser detta problem exakt. Däremot kan man hitta ganska bra, men inte optimala, lösningar på rimlig tid. Precis som i fallet med kontinuerliga problem finns det alltså en *lättlöst* klass av problem och en *svårlöst*. För diskreta problem heter den lätta klassen **P** och den svåra **NP-svår**. **P** står för polynomiell, dvs att lösningstiden som funktion av problemstorleken är ett polynom. **NP** står för *nondeterministic polynomial*, och **NP-svår** är en teoretiskt komplicerad klass, men sammanfattningsvis kan man säga att en exponentialfunktion beskriver lösningstiden som funktion av problemets storlek. Att det senare är avsevärt mycket sämre framgår av tabell 4.1 nedan.

Problemstorlek:	Polynomiel tid (**P**): x^2	Exponentiell tid (**NP-svår**): 2^x
$x = 10$	$x^2 = 100$	$2^x = 1024$
$x = 20$	$x^2 = 400$	$2^x = 1048576$
$x = 40$	$x^2 = 1600$	$2^x \approx 1.1 \times 10^{12}$
$\vdots$	$\vdots$	$\vdots$

Tabell 4.1: Exempel på hur beräkningstiden kan bero på problemstorleken för ett problem i **P** respektive **NP-svår**. Notera att för $x = 40$ tar lösningen för problemet i **NP-svår** ca en miljard gånger längre tid att beräkna.

Inom diskret optimering finns flera problemklasser än **P** och **NP-svår**, men dessa två är de viktigaste att känna till.

För problem i **P** finns ingen generell metod, som *steepest decent*-släktingarna till konvexa problem, utan varje problemtyp måste lösas för sig.

4.4 Att skilja svåra optimeringsproblem från lätta

För både kontinuerliga och diskreta problem har vi nu sett exempel på samma fenomen. Det finns två stora kategorier av problem, de som tar riktigt lång tid att lösa, och de som går ganska snabbt. Och det är *inte* så att man kan vara lite smart och lösa ett **NP-svårt** problem snabbt. Skulle någon lyckas lösa ett sådant problem skulle man nämligen kunna lösa alla, ett sensationellt resultat, som framgår av citatet nedan.

> The class of NP-complete problems are all widely considered unsolvable by polynomial algorithms. [3, sid 3].

Notera att NP-komplett är en delmängd av **NP-svår**. För kontinuerliga problem är gränsen inte lika skarp, konvexa problem är alltid lätta, men det finns icke-konvexa som också är det. Generellt är det emellertid här gränsen går, se nedan.

> In fact the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity, [4, sid 16].

Vi sammanfattar resultaten i följande tabell.

	Kontinuerliga problem	Diskreta problem
Lättlösta	Konvexa	**P**
Svårlösta	Icke konvexa	**NP-svåra**, Exponentiella

Tabell 4.2: Både kontinuerliga och diskreta problem kan delas upp i svårlösta och lättlösta.

När vi i tabellen skriver att problemen är svårlösta innebär det att man i praktiken får välja mellan antingen extremt långsamma algoritmer, eller algoritmer som hittar bra, men inte optimala, lösningar på kort tid. Detta motiverar fråga 4 i kapitel 2. Exempel på algoritmer som används för ickekonvexa och **NP-svåra** problem är de heuristiker som beskrivs i avsnitt 5.2 och 5.3.

5 Att se AI-metoder som Optimeringsmetoder

Denna rapport är för kort för att i detalj redogöra för AI algoritmer, men för att exemplifiera kopplingen mellan AI och optimeringslära skall vi ytligt beskriva några metoder. Slutligen skall vi se vad som menas med *lärande system* inom AI, och motivera frågorna under punkt 5 i kapitel 2.

5.1 Genetiska algoritmer

Genetiska algoritmer är optimeringsalgoritmer som hämtat inspiration från naturens evolutionsprocess, så som den beskrivs av Darwin. I naturen finns ett antal individer inom varje art, och varje individs framgång bestäms av dess genetiska anlag. Individerna konkurrerar, och de med bäst anlag får mer avkomma än de med sämre anlag. På så sätt kommer nästa generation att vara bättre anpassad till de livsbetingelser som råder. Förutom den blandning av anlag som sker vid parning sker även mutationer. Då ändras ett anlag från en generation till nästa på ett slumpmässigt sätt. Detta fenomen gör att framgångsrika anlag kan uppstå, utan att redan finnas i gruppens samlade anlag.

Exempel 5.1 *I praktiken kan man göra så att man kodar de möjliga lösningarna i binär form, som en sträng ettor och nollor, [5, p 619].*

Vi börjar med en population bestående av tre individer: $a_1 = [000111]$, $a_2 = [101111]$, $a_3 = [100011]$. *För att se hur bra de klarar sig evaluerar vi målfunktionen* f *som vi här inte definierar mer än att* $f(a_1) = 7$, $f(a_2) = 5$, $f(a_3) = 2$. a_3 *klarar sig sämst och tas därför bort ur populationen. Till nästa generation låter vi* a_1 *och* a_2 *blanda sina anlag, vi får* $b_1 = [100111]$, *där vi för varje position slumpat om vi tagit genen från* a_1 *eller* a_2. *Sedan gör vi en mutation,* $b_2 = [100110]$. *Notera att sista nollan aldrig hade kunnat uppstå genom parning, eftersom ingen tidigare individ har en nolla i den sista genen. I nästa utvärdering har vi alltså populationen* a_1, a_2, b_1, b_2. *Så fortsätter algoritmen med utvärdering, bortsortering, parning och mutationer, tills ett önskat antal generationer passerat, eller tills förbättringstakten avtagit under någon given gräns. Den bästa individen används sedan som lösning.*

Som synes följer algoritmen den allmänna principen för heuristiker som beskrevs i avsnitt 4.3.1, med slumpvis sökning i närheten av bra lösningar. Nu skall vi titta på en annan heuristik.

5.2 Simulated annealing

Simulated annealing har också hämtat inspiration från naturen, men denna gång från metallurgi. Annealing är namnet på det fysikaliska förlopp som sker då man hettar upp och kyler av metaller för att göra dem mer hållfasta genom att öka storleken på den molekylära kristallstrukturen. Genom att värma metallen kan man få molekyler att lämna lokala minima i energi och hitta andra positioner med lägre energi, vilket ger ett starkare material.

Algoritmen fungerar så att den slumpmässigt hittar en lösning i närheten av den gamla. Om den nya lösningen är bättre accepteras den, är den sämre avgör slumpen om den accepteras eller ej, med en sannolikhet baserad på en parameter T (temperaturen). I början av algoritmen låter man T vara stor, för att successivt minska. Förhoppningen är att man genom möjligheten att acceptera försämringar skall kunna hoppa ur lokala minima, [5, p 113]. Detta är återigen en heuristik enligt beskrivningen i avsnitt 4.3.1.

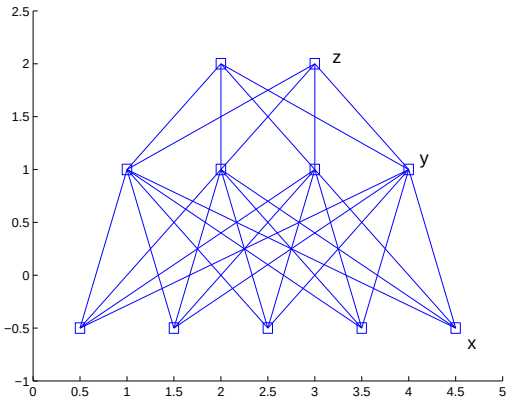
5.3 Tabu Search

Tabu search är till skillnad från de två ovanstående algoritmerna inte inspirerad av fenomen i naturen. Det är istället en rättfram "fix" för att försöka undvika att lokala algoritmer fastnar i lokala min. Iden är enkel, vid varje uppdatering finns en mängd lösningar som inte får väljas, de är Tabu. Den enklaste varianten är när Tabu-mängden helt enkelt är de n senast provade lösningarna (inklusive den nuvarande). Detta gör per definition att algoritmen aldrig stannar, och inte hamnar i cykler som är kortare än n-steg. En mer sofistikerad variant är att Tabu-listan innehåller lösningar som på något sätt påminner om de redan provade. Detta gör emellertid att en oprövad närliggande lösning förblir oprövad. För att i sin tur komma åt detta problem kan man göra undantag för Tabu-regeln, om den nya kandidaten är bättre än den hittills bästa, [6].

Efter att ha betraktat tre exempel på AI-metoder som är rena heuristiska optimeringsalgoritmer skall vi nu se ett exempel på hur lärande kan realiseras genom *steepest decent*-optimering, se avsnitt 4.3.2.

5.4 Lärande system och Neurala Nätverk

Ett neuralt nätverk är en beräkningsstruktur som tar ett givet indata och producerar ett utdata. Strukturen har hämtat inspiration från hur den mänskliga hjärnan är uppbyggd, och dess funktion bestäms av ett antal vikter i det som motsvaras av synapserna, kopplingarna mellan neuronerna. I figur 5.1 visas ett litet exempelnätverk. Varje neuron viktar ihop sina insignaler och avgör sedan med en ickelinjär funktion (g) hur stor signal den skall skicka vidare.

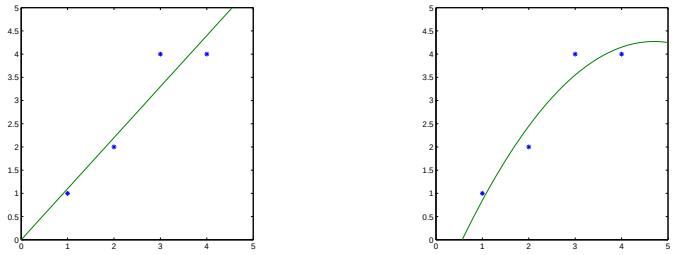


Figur 5.1: Ett exempelnätverk med 5 input-neuroner (x), 2 output-neuroner (z) och ett dolt lager (y).

Formellt kan vi emellertid betrakta hela nätverket som en enda ickelinjär funktion, som bestäms av ett antal parametrar (vikterna), se citatet nedan.

> Notice that because the activation functions g are nonlinear, the whole network represents a complex nonlinear function. If you think on the weights as parameters or coefficients of this function, then learning just becomes a process of turning the parameters to fit the data in the training set, [5, p 572].

Som framgår av citatet handlar alltså lärandet om att välja ett antal parametrar så att en given datamängd approximeras så bra som möjligt. Detta är ett klassiskt kontinuerligt optimeringsproblem, och är principiellt väldigt likt minstakvadratanpassningen av en rät line, se figur 5.2.



(a) $f(x) = k_0 + k_1 x$ (b) $f(x) = k_0 + k_1 x + k_2 x^2$

Figur 5.2: I princip så är lärandet i ett neuralt nätverk helt analogt med hur de två polynomen ovan "lär" sig koefficienterna k_i för att approximera data (*) så bra som möjligt. I båda fallen är det en optimeringsalgoritm som hittar de optimala parametervärdena.

Eftersom funktionen är ickelinjär är lösandet emellertid lite krångligare än i minstakvadratfallet, en populär metod är den *steepest decent*-metod som beskrevs i avsnitt 4.3.2.

> It turns out that the equations can be given a very simple interpretation as a method for performing gradient decent in weight space. [5, p580].

Det är emellertid inte bara i neurala nätverk som lärandet sker i form av lättbegriplig optimering. Det visar sig att alla lärande system handlar om att approximera data med en parametersatt funktion:

> The key point is that all learning can be seen as learning the representation of a function, [5, p529].

Skillnaderna handlar bara om hur denna funktion är representerad, det kan vara allt från polynom till neurala nätverk eller så kallade belief networks.

> The difficulty of learning depends on the chosen representation. Functions can be represented by logical sentences, polynomials, belief networks, neural networks, and others, [5, p558].

Detta motiverar frågorna under punkt 5 i kapitel 2.

I detta kapitel har vi sett flera exempel på hur nyckelfunktionen i AI-metoder kan beskrivas som en optimeringsalgoritm. Därmed återstår bara en kort sammanfattning i sista kapitlet.

6 Sammanfattning

I denna rapport har vi argumenterat för att svaren på de fem frågorna i kapitel 2 ger en snabb, översiktlig och objektiv bild av prestandan hos ett militärt AI-system. Vi har vidare försökt motivera frågorna och gett bakgrund för tolkning av möjliga svar, utan att gå in på alltför djupa teoretiska resonemang.

Den genomgående idéen i rapporten är att AI och optimeringslära har väldigt mycket gemensamt. Vi påstår inte att all AI på ett eller annat sätt involverar optimeringslära, men väldigt ofta går AI för *militära system* att formulera som optimeringsproblem. Detta därför att de flesta uppgifter går att formulera med uppdragstaktik och att alla uppgifter formulerade med uppdragstaktik som är tydliga nog att förstås av en maskin, också kan förstås som optimeringsproblem.

Det finns två fördelar med att se algoritmer som lösare av optimeringsproblem. För det första undviker man att påverkas av Hollywood-visioner och luddiga liknelser med biologiska intelligenta system. För det andra kan man fokusera på objektiva uppgiftsformuleringar i form av bivillkor och målfunktioner, samt förstå och förutsäga prestandabegränsningar utifrån fundamentala resultat i optimeringsläran.

Detta är värdefullt för både upphandling och användning av militära AI-system.

Litteraturförteckning

[1] Försvarsmakten. *Försvarsmaktens Grundsyn Ledning*. ISBN 99-3541815-4, 2001.

[2] D.G. Luenberger. *Linear and Nonlinear Programming*. Kluwer Academic Publishers, 2003.

[3] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, 1998.

[4] R.T. Rockafellar. Lagrange Multipliers and Optimality. *SIAM Review*, 35(2):183–238, 1993.

[5] S.J. Russell and P. Norvig. *Artificial intelligence: a modern approach*. Prentice-Hall, Inc. Upper Saddle River, NJ, USA, 1995.

[6] Wikipedia article on Tabu Search. http://en.wikipedia.org/, Mars 2007.

FOI är en huvudsakligen uppdragsfinansierad myndighet under Försvarsdepartementet. Kärnverksamheten är forskning, metod- och teknikutveckling till nytta för försvar och säkerhet. Organisationen har cirka 1350 anställda varav ungefär 950 är forskare. Detta gör organisationen till Sveriges största forskningsinstitut. FOI ger kunderna tillgång till ledande expertis inom ett stort antal tillämpningsområden såsom säkerhetspolitiska studier och analyser inom försvar och säkerhet, bedömningen av olika typer av hot, system för ledning och hantering av kriser, skydd mot hantering av farliga ämnen, IT-säkerhet och nya sensorers möjligheter.

FOI
Totalförsvarets forskningsinstitut
164 90 STOCKHOLM

Tel: 08-5550 3000
Fax: 08-5550 3100

www.foi.se