

Projektets titel: TALVMENNI ver. 0.1: Design og implementation af en chess engine

Deltagere: Eyðun Lamhauge & Eyðun Nielsen

Vejleder: Kurt Madsen

Baggrund:

En **chess engine** er et program som er stand til at spille skak. D.v.s. finde frem til hvilke træk er lovlige i en given position og ikke mindst, forsøge at finde frem til hvilket træk er det bedste i den givne position. Der til skal dette program kunne administrere sit tidsforbrug til disse beregninger.

Skak tilhører gruppen af spil som kaldes for ”spil af perfekt information” (i modsætning til f.eks. bridge hvor tilfældighed afgør hvilken position man starter på). Hvis nok af tid og plads er til rådighed, så vil en optimal løsning kunne findes ved at lave en fyldegørende søgning blandt alle alternativer. Realiteten er bare den, at løsningsrummet er så stort, at vi hverken har nok af tid eller plads til disse søgninger. Derfor kræves af en chess engine at den er i stand til at optimere sine søgninger, samt evaluere sine resultater.

Forsøget på at lave en maskine der kan spille skak kan dateres helt tilbage til 1769, hvor Baron Wolfgang de Kempelen konstruerer hvad han påstår er en talvspillende automat kaldet ”The Turk”. Men denne og nogle flere senere hen i det 19. århundrede viste sig alle at være fupnumre, hvor der var menneskelige skakspillere gemt indeni.

Men i 1951 beskriver Alan Turing (der dog selv ikke var nogen skrap skakspiller) det første program der kan spille et helt spil skak.¹ I slutningen af 60’erne kunne skakprogrammerne hamle op med skakspillere på nybegynder-niveau. Omkring 80’erne var de lige så gode som en menneskelig skak-spiller på ekspert-niveau. I starten af 90’erne var de mesterspillere, og det kulminerede i 1997 da IBM's skakcomputer ”Deep Blue” vandt en match over verdensmesteren Garry Kasparov, der af flere regnes for at være den stærkeste skakspiller igennem alle tider. (Matchens omstændigheder er dog senere blevet stærkt kritiseret for at favorisere maskinen).

I dag findes der omkring 200-300 chess engines. Der fleste er udviklede af glade amatører og så nogle få stærke kommercielle produkter. Der arrangeres hvert år både nationale mesterskaber samt verdensmesterskabet for at finde den stærkeste chess engine. Det ellefte verdensmesterskab blev holdt i Sveits i November 2003 og vinderen blev det tysk-udviklede Shredder efter omkamp imod sin landsfælle Deep Fritz.

¹ http://en.wikipedia.org/wiki/Computer_chess

Formål med projektet:

Projektets formål er at designe og implementere en **chess engine** og at udarbejde et rapport som belyser arbejdet og de overvejelser der er indgået.

- En chess engine indeholder efter vores mening flere interessante og udfordrende discipliner inden for matematik og datalogi, så som søgning, evaluering, evt. AI, o.s.v.
- Vores undersøgelser indtil videre indikerer at de fleste nuværende chess engines er designede på et meget maskinnært niveau med henblik på høj ydelse. Vi vil undersøge om det er muligt at designe og implementere en højt ydende chess engine der samtidigt er implementeret objekt-orienteret i Java med den fleksibilitet som dette giver.

Løsningsforslag:

- Man kan opdele skakprogrammet i to komponenter. En chess engine (defineret i afsnittet baggrund) og en grafisk brugergrænseflade. En GUI har til opgave at tegne skakbrættet på skærmen og reagere på input fra brugeren og fra chess engine. Vi vil ikke udvikle vores egen GUI, men bruge en GUI som allerede eksisterer på markedet. For at bruge en standard-GUI kræves at kommunikationen imellem vores engine og GUI'en overholder en allerede defineret protokol. To protokoller bliver hovedsageligt brugt i dag, Winboard og UCI. Winboard er mest udbredt, da det er mere end 10 år siden at den først blev defineret. UCI (Universal Chess Interface Protocol) er af nyere dato, og har specielt vundet indpas hos de kommercielle chess engines.
Vi vil undersøge begge alternativer og vælge den protokol som egner sig bedst til vores formål.
- Man bliver nødt til at finde en måde at repræsentere en given position på brættet internt i memory. To metoder bliver mest brugt i dag, vektor-metoden og bitboards. Vi vil kort belyse vektor-metoden, men specielt undersøge bitboard-metoden nærmere, da vi allerede har valgt at bruge bitboards til vores implementation. Bitboard-metoden blev oprindeligt opfundet i soviet for mange år siden, men er ikke så udbredt som vektor-metoden, da den nok er lidt sværere at forstå og abstrahere. Vi mener at bitboards egner sig bedre til vores engine, da vores implementation bliver udviklet i Java, som har brug for den ekstra ydelse som bitboards giver over vektor-metoden, samt regner vi med at vi med en god objekt-orienteret abstraktion kan indkapsle evt. kompleksitet som brug af bitboards medfører.
- Vi vil undersøge i hvilket omfang vi kan prekalkulere bitmap-repræsentationer for attacks, moves, king-safety m.v. og lægge disse i Hash-tabeller således at vi kan udnytte den forholdsvis store hukommelse i dagens computere og derved få en bedre ydelse.
- Flest alle chess engines i dag, arbejder med at søge igennem et træ af alle mulige træk og modtræk. Roden er den aktuelle position og hver gren repræsenterer et muligt lovligt træk. Ved hvert af bladene vil en evalueringsfunktion tildele en værdi til positionen, og chess engine vil så vælge den sti med den højeste værdi. Vi vil undersøge hvilke søgningsmetoder der eksisterer, og implementere vores egen søgningsfunktion. Samt vil vi undersøge og implementere en evalueringsfunktion som har til opgave at vurdere skakpositioner.
- Vi har valgt at implementere TALVMENNI i Java, fordi vi ønsker at se om hvorvidt vi i dag kan få sådan en implementation til at køre hurtigt nok til at få en stærk chessengine. Java har haft et dårlig ry med hensyn til ydelse, hvilket nok også var berettiget for nogle år siden. Men hvis vi kan få presset nok ydelse ud af TALVMENNI på en enkelt Java VM (og vi har den fornødne tid), så vil vi undersøge om vi kan nå at få udnyttet den fleksibilitet der er i Java med mobil kode m.v. ved at lave en distribueret version af TALVMENNI kørende på Jini/JavaSpaces. Derfor vil vi også i vores design af TALVMENNI være opmærksomme på hvor vi kan udføre parallelle beregninger, som vi vil kunne udnytte senere i en distribueret version.

- Sun's Java VM har siden 1997 indholdt en Just-in-time (JIT) compiler, også kaldet HotSpot compiler. Det som denne gør er, at under afviklingen af et Java program, vil VM'en holde øje om der er metoder der bliver kaldt rigtig meget, dvs. bliver kaldt flere gange end en tærskelværdi (f.eks. 1500 som standard), så er disse hotspots i programmet og VM'en aktiverer JIT-compileren, der compilerer disse metodens bytecode til native code dvs. maskinkode. Der er sket en del udvikling og forbedringer på denne JIT-compilerings teknologi siden 1997. Og vi vil selvfølgelig prøve at udnytte dette i TALVMENNI ved at vi løbende profilerer vores kode, således at vi kan se hvilke metoder, der evt. Kan 'varmes' op i forbindelse med opstart.
- Indenfor skakkens verden beregnes skakspillernes styrke efter et såkaldt ratingsystem. Dette system beregner et ratingtal ud fra spillerens resultater imod andre spillere og ud fra hvilke ratingtal disse modstandere har. Dette system bliver ligeledes brugt til at finde spillestyrken for skakprogrammet. Vi vil hurtig færdiggøre den første alpha-version af vores chess-engine, for at lægge den ud på internettet at spille imod både mennesker og andre chess-engines. Dette vil efter kort tid give vores chess-engine et ratingtal, som vi så efterfølgende kan følge udvikling af, så hvert vi laver ændringer og forbedringer.

Arbejds- og tidsplan:

Vi har valgt at udarbejde vores arbejds- og tidsplan, således at vi får nogle regelmæssige milestones i projektforsløbet.

1. Vi har valgt at disse iterationer skal være af 6 ugers varighed, dog undtagen M1-M2 og M2-M3, hvor varigheden er sat til 8 uger pga. at eksamen i IT-Strategi falder i den første og sommerferie i den anden.
2. Det andet, som vi har valgt, er at definitionen af hvad det bliver, som vi afleverer til de enkelte milestones, bliver udspecificeret en til to milestones ad gangen, da vi ikke mener at selve planen er det væsentlige, men det at planlægge og derfor ville det være uhensigtsmæssigt hvis planen blev færdig... For at citere Eisenhower: "In preparing for battle I have always found that plans are useless, but planning is indispensable"²
3. Og som det tredje skal det som vi afleverer til hver milestone være målbart.

- 31-03-2004 **M1**: Udforming af projektplan (dokumenteret i nærværende dokument). Programmering påbegyndes med det samme. Indehold for M2 defineres og M3 defineres delvis.
- 26-05-2004 **M2**: Research og litteratursøgning afsluttes. Endelig projektformulering udarbejdes og dokumenteres. Indehold for M3 og M4 uddybes.
- 21-07-2004 **M3**: Alpha-release af vores chess engine med tilhørende dokumentation - [uddybes senere].
- 01-09-2004 **M4**: Analyse af testresultaterne med alpha-maskinen – [uddybes senere].
- 13-10-2004 **M5**: beskrivelse af konsekvenserne for den endelige maskine [uddybes senere].
- 24-11-2004 **M6**: Afslutning - [uddybes senere].

2 Richard Nixon, *Six Crises*(New York: Touchstone Press, 1990)

Referencer:

Der findes desværre ikke megen tilgængelig litteratur omkring emnet "chess-engine" i bogformat. En søgning på emnet på amazon giver 45 resultater, men alle bøgerne er udgået fra forlaget! Men i stedet, så findes der en hel del materiale på internettet.

Shannon, Claude E. "Programming a computer for playing chess": Philosophical Magazine, Vol. 41, 1950.

Klassisk læsning indefor computerprogrammering.

<http://www.tim-mann.org/xboard/engine-intf.html>: Chess Engine Communication Protocol.

En kommunikationsprotokol som vi kommer til at bruge i forbindelse med kommunikationen imellem vores engine og Xboard-brugergrænsefladen (Hvis det bliver xboard, der bliver vores GUI).

<http://www.gnu.org/software/chess/chess.html>: Chess GNU Project.

GNU Chess er middel-stærk open-source chess-engine som vi uden tvivl vil studere en hel del nærmere.

<http://www.cis.uab.edu/info/faculty/hyatt/hyatt.html>

Crafty er en meget stærk open-source chess-engine som er udviklet af Robert Hyatt. Blandt andet så vandt dens forgænger "Cray Blitz" verdensmesterskabet for chess engines i 1983 og 1986.

<http://www.cis.uab.edu/info/faculty/hyatt/pubs.html>

Hyatt har skrevet nogle artikler omkring emnet som er relevant læsning.

<http://www.talkchess.com/forums/1/index.html>

Message board hvor skakprogrammører mødes for at diskutere.

<http://www.xs4all.nl/~verhelst/chess/programming.html>

Paul Verhelst hjemmeside, hvor der findes materiale om computer-skak programmering, samt en del links til andre hjemmeside om emnet.

<http://wbec-ridderkerk.nl/>

Ejeren af denne hjemmeside Leo Dijksman, arrangerer løbende turneringer for chess engines, hvor han har opdelt dem i divisioner efter spillestyrke.