Examensarbete

HÖGSKOLAN I SKÖVDE

EN FLEXIBEL SCHACK AI BASERAD PÅ CASE-BASED REASONING

A CASE-BASED REASONING APPROACH TO A FLEXIBLE CHESS AI

Examensarbete inom huvudområdet Datavetenskap
Grundnivå 30 högskolepoäng
Vårtermin 2015

Johannes Qvarford

Handledare: Peter Sjöberg
Examinator: Anders Dahlbom

Sammanfattning

(Ta bort irrelevanta rubriker, eller låta dem vara tomma?)

Innehållsförteckning

1 Introduktion 1

2 Bakgrund 2

2.1 Case-based Reasoning och Case-based Planning 2

2.2 Schack 2

2.2.1 Regler 2

2.2.2 Elo-rating 6

2.2.3 Portable Game Notation 6

3 Problemformulering 8

3.1 Problembeskrivning 8

3.2 Metodbeskrivning 8

3.2.1 Generera falldatabaser 8

3.2.2 AI-agenten 8

4 Genomförande/Implementation/ Projektbeskrivning 9

5 Utvärdering 10

6 Avslutande diskussion 11

Referenser 12

1 Introduktion

Schack har en lång historia som sträcker sig ända till början av 600-talet e.Kr.. Schack har traditionellt spelats mellan två människor, men under det senaste århundradet har även maskiner utvecklats för att spela spelet. Dessa maskiner har visat sig kunna mäta sig med mänskliga spelare, och redan under 70-talet utvecklades en maskin som kunde besegra en stormästare (Hapgood 1982). Sedan dess har utvecklingen bara fortsatt, och 1997 besegrades den då regerande världsmästaren Garri Kasparov av en schackspelande maskin vid namn *Deep Blue* (Cambell, Hoane & Hsu 2001).

Schackmaskiner har förbättrats genom historien, men inte genom att efterlikna människor. Hapgood (1982) påstod att schackmaskinerna spelade fult, men vann genom att utnyttja små misstag som motståndaren gjorde. Inte mycket forskning har gjorts kring området att skapa realistisk artificiell intelligens (AI) som kan spela schack på olika skicklighetsnivåer. För att tackla detta område presenteras en AI-agent som använder *Case-Based Reasoning* (CBR) för att efterlikna mänskligt beteende. CBR är en teknik för att utveckla AI-agenter, som bygger på att lösa problem baserat på lösningar av tidigare, liknande problem. När en agent ska göra ett drag kan den härma vad en expert gjort i samma läge, genom att konsultera en fallbas. Alla möjliga fall kan inte lagras eftersom det enligt Shannon (1950) lär finns uppemot 10^{54} fall i schack. AI-agenten måste därför ibland basera sitt val på det expertläge som är mest likt det nuvarande läget.

I det här arbetet ska ett program av en schackspelande AI-agent skapas som använder CBR. Agenten ska kunna använda falldatabaser från matcher spelade av spelare av olika rank, för att justera dess svårighetsgrad. Svårighetsgraden ska överensstämma med spelarnas rank, så att agenten är svårare att besegra om dess falldatabas är baserad på högt rankade spelare än om den är baserad på lågt rankade spelare. Falldatabaser kommer kunna skapas baserade på tidigare spelade matcher dokumenterade i *Portable Game Notation* (PGN); ett format som används av bl.a. World Chess Federation (FIDE) (http://www.fide.com). Arbetet ska kunna användas som grund för att utveckla AI-agenter för spel med justerbar svårighetsgrad.

(När jag refererar till FIDE som organisation, ska jag ha med länken eller inte?)

2 Bakgrund

I denna sektion presenteras bakgrundsinformation till CBR och hur det tidigare har applicerats inom forskning. Här visas även spelet schack, dess regler, hur schackspelare rankas och PGN - det vida använda formatet för att beskriva schackmatcher.

2.1 Case-based Reasoning

CBR är en teknik för problemlösning som är baserad på idén att använda lösningar på tidigare, liknande problem. Det går ut på att låta en artificiell eller mänsklig expert lösa ett problem, och dokumentera hur den valde att göra det. Lösningarna på problemen från experten behöver inte nödvändigtvis uppfylla några korrekthetskrav. Ett problem tillsammans med sin lösning bildar ett fall, och en grupp fall kallas för en fallbas. När en annan agent ska lösa ett problem kan den härma hur en expert löste problemet eller ett liknande problem genom att konsultera en fallbas (Richter & Weber 2013).

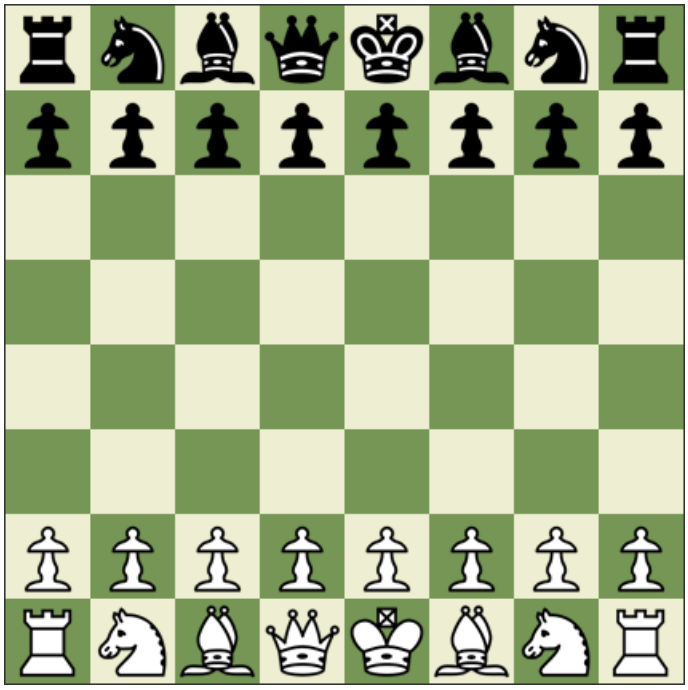
Nära besläktat med CBR är tekniken *Case-Based Planning* (CBP) som går ut på att basera planer på tidigare lyckade planer och återhämtningar från misslyckade planer. CBP innefattar mer än att bara hitta en lösning och utföra den, som i CBR. En plan består av ett antal handlingar som är delvis ordnade, dvs. vissa handlingar måste ske efter andra, men ordningen av gruppmedlemmar inom vissa grupper av handlingar spelar ingen roll. Inom CBP utvecklas planbaser organiskt; när en plan har hämtats, anpassats för problemet och utförts så sparas den i planbasen. Med CBP representeras problem som förvillkor, och planer har invarianter som måste gälla när de utförs, annars avslutas/avbryts dem (Spalazzi 2001).

CBR och CBP har applicerats på datorspel i ett antal tidigare arbeten. Rekabdar, Shadger och Osareh (2012) presenterar hur CBP kan användas för att lära agenter att spela ett fotbollsspel tillsammans genom att låta dem observera hur ett motståndarlag spelar. I (Aha & Molineaux & Ponsen 2005) visas ett sätt att hantera den stora mängd fall som uppstår i Warcraft II (Blizzard 1995) modden Wargus (The Wargus Team 2002) när CBP appliceras. Ontañón, Mishra, Sugandh och Ram (2007) studerar även hur CBR kan appliceras på Wargus, men i det arbetet ligger fokus på hur annoterade fall kan användas för att hitta beroenden mellan mål, och resonera fram planer för att nå dessa mål. Fallen annoterades i arbetet med de mål som experten försökte uppnå i varje fall när den agerade som den gjorde. Det turbaserade strategispelet (TBS) Call to Power || (CTP2) användes som provunderlag i arbetet av Sanchez-Ruiz et. al (2007). I arbetet undersöks hur sökningen av relevanta fall i CBR kan förbättras genom att bara lagra den del av spelläget som är relevant för att utföra handlingen i ett visst fall.

2.2 Schack

2.2.1 Regler

Schack är ett turbaserat brädspel för två spelare där målet är att besegra sin motståndare. Spelet utspelar sig på en 8x8-rutors spelplan, där varje spelare kontrollerar varsin armé av spelpjäser – vit och svart (FIDE u. å.) **(Föregående citat är till hela den här sektionen. Ska jag indikera det på något speciellt sätt, eller behövs det tom ingen referens till schack? Om källan innehåller alla regler och mer än jag tar upp här, behövs den här sektionen ens?)**. I Figur 1 visas en bild av spelplanen i början av spelet.



Figur 1 Bild av spelplanen i början av spelet.

Spelarna turas om att flytta spelpjäser i sina arméer. En spelare får bara flytta en spelpjäs per drag. Två pjäser av samma färg får inte ockupera samma ruta. Om en spelare flyttar en av sina spelpjäser på en ruta ockuperad av en motståndarpjäs, så fångas motståndarpjäsen och lämnar spelplanen för resten av matchen. Högst en pjäs i taget får ockupera en ruta, och en pjäs får generellt inte flytta till en ruta om andra pjäser står i vägen till rutan. Om en pjäs kan flytta till specifik ruta, så betraktas det som att pjäsen hotar rutan, eller att pjäsen som står på rutan hotas.

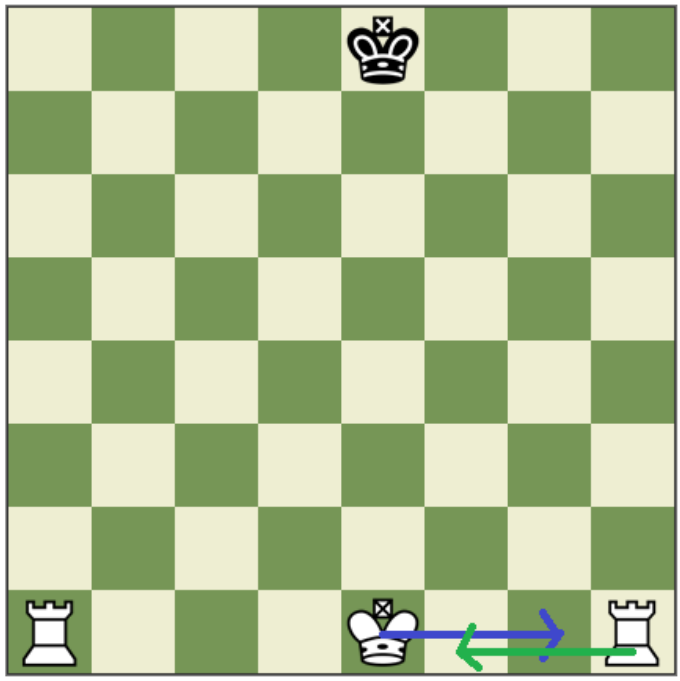
Bonden (♟) kan flytta sig ett steg rakt framåt (sett från den ägande spelarens håll), eller ett steg diagonalt framåt om draget är ett fångande drag. Springaren (♞) kan flytta sig två steg horisontellt eller vertikalt, och ett steg på den resterande axeln. Springaren kan flytta till en ruta även om det finns pjäser som blockerar vägen. Löparen (♝) kan röra sig diagonalt. Tornet (♜) kan röra sig horisontellt eller vertikalt. Drottningen (♛) kan antingen röra sig horisontellt, vertikalt eller diagonalt. Kungen (♚) kan röra sig ett steg horisontellt, vertikalt eller diagonalt.

Om en bonde når den sista raden sedd ur ägarens perspektiv, så kan den omvandlas till vilken annan pjäs som helst.

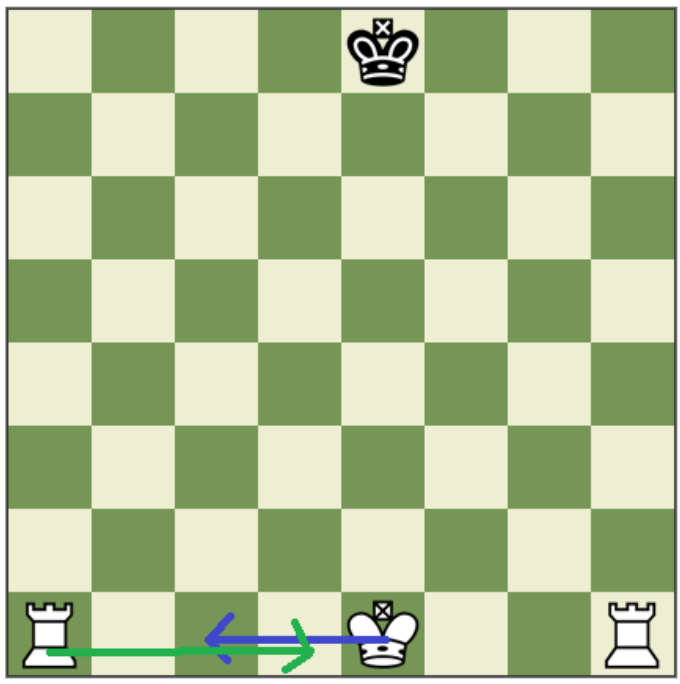
Bonden kan flytta två steg rakt framåt på sitt första drag. Om en spelare flyttar en bonde två steg, så kan bonden betraktas som om den bara tog ett steg, om den fångas av en motståndarbonde nästa drag. Detta kallas *en passant* och illustreras i Figur 2.

Figur 2 Bild av *an passant*. Om vit flyttar sin bonde två rutor framåt kan den svarta bonden fånga den genom att flytta till rutan som den röda pilen indikerar.

En spelare kan göra så kallad rockad med sin kung och ett torn, om det inte finns några pjäser mellan tornet och kungen, och varken tornet eller kungen har flyttats förut. Utöver det får kungen inte vara hotad, fördas över några rutor som hotas och dess slutgiltiga ruta får inte heller vara hotad. Rockaden går till så att kungen flyttas två steg i tornets riktning, och tornet flyttas i kungens riktning så att den hamnar en ruta på andra sidan av kungens nya position. Figur 3 illustrerar hur detta kan se ut om den vita kungen gör rockad med det närmaste tornet, och Figur 4 illustrerar rockad med tornet längst bort.

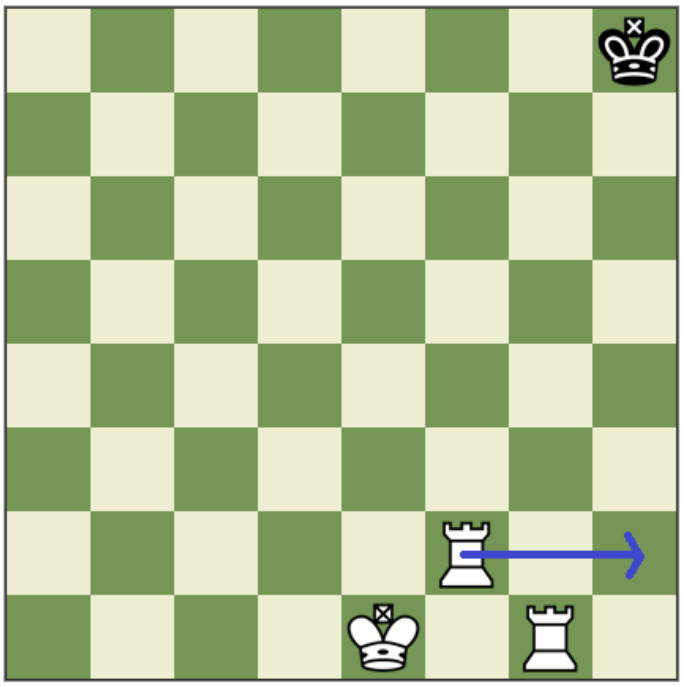


Figur 3 Bild som visar hur pjäserna flyttas när vit gör kort rockad.



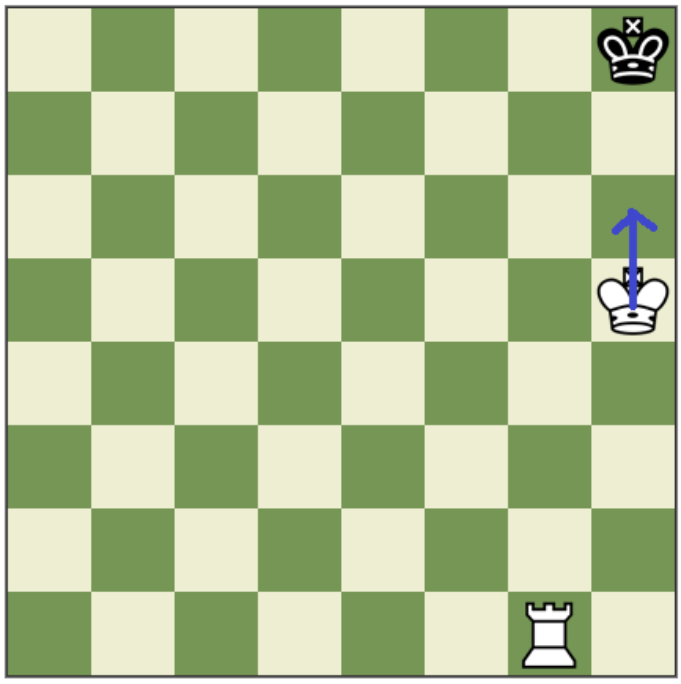
Figur 4 Bild som visar hur pjäserna flyttas när vit gör lång rockad.

En spelare får aldrig göra ett drag som leder till att motståndaren hotar spelarens kung. Om en spelare gör ett drag så att motståndarens kung hotas kallas det för schack. Detta gäller även om kungen hotas av pjäser som skulle lämna sin kung hotad om de flyttade från sin ruta. Om motståndaren inte kan följa upp med ett drag som försätter kungen ur schack vinner spelaren, vilket kallas för schack matt. Ett exempel av schack matt visas i Figur 5.



Figur 5 Bild som visar hur vit kan göra schack matt. Kungen hotas av tornet på andra raden, samtidigt som den inte kan flytta sig utan att fortfarande hotas.

Om motståndarens kung inte hotas, men samtidigt inte kan göra något drag utan att kungen hotas så blir det lika, vilket även kallas för patt eller remi. Ett exempel av patt visas i Figur 6.



Figur 6 Bild som visas hur vit kan göra patt. Den svarta kungen hotas inte, men samtidigt kan den inte flytta sig någonstans utan att hotas av tornet eller den vita kungen.

2.2.2 Elo-rating

Elo-rating är ett sätt att ranka schackspelare relativt till varandra. I Elo-rating rankas spelare i form av poäng. En spelare som besegrar en annan spelare ökar i rank, medan den andras rank minskar. En spelare ökar mindre i rank när den besegrar en spelare med lägre rank, än när den besegrar en högre rankad spelare. Likaså minskar en spelares rank mer när den förlorar mot en lägre rankad spelare, än mot en högre rankad spelare. Om det blir remi mellan två spelare, går spelaren med högst rank ner i rank och motspelaren går upp i rank (FIDE 2012).

2.2.3 Portable Game Notation

PGN är ett format som utvecklades för att spara och beskriva schackmatcher (Huber 2006). Ett PGN-dokument kan innehålla ett antal matcher, och varje match innehåller metainformation om matchen, och dragen som utfördes i matchen. Informationen kan gälla när/var matchen spelades och av vilka. Dragen skrivs med algebraisk notation (AN).

AN är en notation som beskriver drag kortfattat till den grad att de inte är tvetydiga. Raderna numreras från vits håll med bokstäver från a till h, och kolumnerna med siffrorna 1 till 8. En position på spelplanen kan då beskrivas med dess tillhörande rad och kolumn t.ex. e4 eller a2. Ett drag har ett prefix med stor bokstav som beskriver vilken sorts pjäs som flyttades. N för springare, R för torn, B för löpare, Q för drottning, K för kung, medan bonde saknar prefix. Detta följs av positionen som pjäsen flyttades till. Exempel: Ke1, Nf3, c4. Om draget är ett fångande drag så sätts ett x framför positionen som pjäsen flyttades till. De drag som leder till schack har ett plustecken som suffix. Rockad med närmaste torn representeras med ”O-O” och rockad med torn längst bort representeras med ”O-O-O”.

I de fall då ett drag är tvetydigt, t.ex. om två springare på e4 respektive e6 kan flytta till c5, så följs pjäsbokstaven av radkoordinaten eller kolumnkoordinaten beroende på vilken som kan

uttrycka draget unikt (Nee5 är inte unikt i detta fall, medan N4e5 är det). Efter det sista draget i matchen visas resultatet 1-0, 0-1, eller ½-½ om vit vann, förlorade, respektive gjorde remi med svart. I Figur 7 visas ett exempel av en match beskriven i PGN.

```
[Event "F/S Return Match"]
[Site "Belgrade, Serbia Yugoslavia|JUG"]
[Date "1992.11.04"]
[Round "29"]
[White "Fischer, Robert J."]
[Black "Spassky, Boris V."]
[Result "1/2-1/2"]

1. e4 e5 2. Nf3 Nc6 3. Bb5 a6 {This opening is called the Ruy Lopez.}
4. Ba4 Nf6 5. O-O Be7 6. Re1 b5 7. Bb3 d6 8. c3 O-O 9. h3 Nb8  10. d4 Nbd7
11. c4 c6 12. cxb5 axb5 13. Nc3 Bb7 14. Bg5 b4 15. Nb1 h6 16. Bh4 c5 17. dxe5
Nxe4 18. Bxe7 Qxe7 19. exd6 Qf6 20. Nbd2 Nxd6 21. Nc4 Nxc4 22. Bxc4 Nb6
23. Ne5 Rae8 24. Bxf7+ Rxf7 25. Nxf7 Rxe1+ 26. Qxe1 Kxf7 27. Qe3 Qg5 28. Qxg5
hxg5 29. b3 Ke6 30. a3 Kd6 31. axb4 cxb4 32. Ra5 Nd5 33. f3 Bc8 34. Kf2 Bf5
35. Ra7 g6 36. Ra6+ Kc5 37. Ke1 Nf4 38. g3 Nxh3 39. Kd2 Kb5 40. Rd6 Kc5 41. Ra6
Nf2 42. g4 Bd3 43. Re6 1/2-1/2
```

Figur 7 En schackmatch i PGN-formatet. Notera att numreringen inte ökar för varje drag, utan varje par av drag.

3 Problemformulering

3.1 Problembeskrivning

Syftet med arbetet är att skapa ett program av en schackspelande AI-agent. Programmet ska kunna konfigureras innan det körs med en svårighetsgrad, och baserat på svårighetsgraden ska en fallbas byggas som AI-agenten ska referera till när den ska bestämma vilket drag den ska utföra. Ett hjälpprogram ska skapas för att skapa falldatabasfiler från grupper av schackmatcher dokumenterade i PGN.

3.2 Metodbeskrivning

3.2.1 Generera falldatabaser

För att kunna skapa stora och varierade fallbaser måste en stor mängd data samlas in från experter. Det finns flera schackmatchdatabaser som innehåller många matcher sparade i PGN, och i det här arbetet så kommer matcher från FIDE:s databas att användas (http://ratings.fide.com/).

Att tolka PGN under körtid kan ta en del processorkraft som hellre bör användas för AI-agentens beslut. För att komma runt detta kan ett antal PGN-matcher konverteras till en fallbasfil i förväg, som är lättare och snabbare att använda av AI-agenten, och dessutom kan användas flera gånger. En fallbas kommer innehålla ett antal lägen, och de drag som utfördes i respektive läge. Programmet kommer att använda fall från matcher spelade av högre rankade spelare vid högre svårighetsgrader.

Att översätta från ett format till ett annat som sedan måste tolkas under körtid kan ifrågasättas, eftersom det introducerar ett till synes onödigt steg i processen att tolka de redan sparade matchfilerna. Uppdelningen är inte bara av effektivitetsskäl, men fokus. Att separera dem öppnar upp möjligheten att skapa fallbasfiler på andra sätt, och skapandet av andra schackspelande AI-agenter som använder fallbasfiler.

3.2.2 AI-agenten

AI-agentens uppgift är att bestämma ett drag att utföra i ett visst läge, givet dess fallbas. Den uppfyller detta genom att undersöka alla fall i listan, och göra det drag i fallet vars läge är mest likt det givna läget, och vars drag går att utföra i det givna läget.

Programmet ska kommunicera med *Universal Chess Interface* (UCI) (Rupert, 2006), vilket är ett kommunikationsprotokoll mellan schackmotorer och användargränssnitt. Ett alternativt kommunikationsprotokoll vid namn XBoard (Mann & Muller 2009) har övervägts, men har valts bort på grund av att det har mindre stöd och ekosystem än UCI. Det grafiska användargränssnittet (GUI) Arena Chess som stödjer UCI ska användas för att testa AI-agenten mot sig själv, och se om den presterar bättre på högre svårighetsgrader, med olika fallbaser.

Att använda ett vida använt protokoll som UCI gör det inte bara kompatibelt med flera olika användargränssnitt för schack, men öppnar upp möjligheten att använda AI-agenten mot andra agenter. Det här arbetet kommer inte jämföra AI-agenten mot andra agenter, då en stor del av dessa är fokuserade på att spela optimalt och syftet med arbetet är inte att skapa en så smart AI-agent så möjligt, utan en som kan anpassa sig till en svårighetsgrad.

4 Genomförande/Implementation/ Projektbeskrivning

5 Utvärdering

6 Avslutande diskussion

Referenser

Activision (2000). *Call to Power II* (Version 1.0) [Datorprogram]. Activision http://www.gog.com/game/call_to_power_2

Aha, D. W., Molineaux M. & Ponsen M. (2005). Learning to Win: Case-Based Plan Selection in a Real-Time Strategy Game. *Case-Based Reasoning Research and Development,* ss. 5-20.

(Warcraft II finns inte tillgängligt officiellt längre på någon hemsida)

Blizzard (1995). *Warcraft* || (Version: 1.0) [Datorprogram]. Blizzard.

Campbell, M., Hoane, A. J. & Hsu, F. H. (2002). Deep Blue. *Artificial intelligence*, 134(1), ss. 57-83.

(Referensen nedan är för ett veckomagasin, hur ska jag referera där?)

Hapgood, F. (December 1982 23/30). Computer Chess Bad - Human Chess Worse. *New Scientist*, 96(1337), ss. 827–830.

Huber, R. (2006). *Description of the universal chess interface (UCI).* http://download.shredderchess.com/div/uci.zip [2015-02-08]

Mann, T. & Muller, H. G. (2009). *Chess Engine Communication Protocol.* http://www.gnu.org/software/xboard/engine-intf.html [2015-02-09]

(De två referenserna nedan saknar volume+issue för att jag inte kunde hitta det.)

Ontañón, S., Mishra, K., Sugandh, N. & Ram, A. (2007). Case-Based Planning and Execution for Real-Time Strategy Games. *Case-Based Reasoning Research and Development,* ss. 164-178.

Rekabdar, B., Shadger, B. & Osareh, A. (2012) Learning Teamwork Behaviors Approach: Learning by Observation Meets Case-Based Planning. *Artificial Intelligence: Methology, Systems, and Applications,* ss. 195-201.

Richter, M. M., & Weber, R. O. (2013) *Case-Based Reasoning: A Textbook* (s. 17-26). Berlin: Springer-Verlag.

(Väldigt osäker på referensen nedan. Först kan jag inte hitta någon publikationsinformation, så jag refererar den som en rapport. Jag kan heller inte hitta någon huvudkälla, då rapporten refererar till flera universitet, delvis stöds av flera organisationer, och så lagras PDF:en under ett annat universitets domän. För tillfället använder jag ägaren till domänen, dvs. Georgia Tech.)

Sanchez-Ruiz, A., Lee-Urban, S., Muñoz-Avila, H., Díaz-Agudo, B. & González-Calero P. (2007). *Game AI for a Turn-based Strategy Game with Plan Adaptation and Ontology-based retrieval.* Georgia Tech. http://www.cc.gatech.edu/~surban6/publs/ICAPS-PG07.pdf [2015-02-08]

(Saknar sidor.)

Shannon, C. E. (1950). Programming a Computer for Playing Chess. *Philosophical Magazine*, 41(314).

Spalazzi, L. (2001). A Survey on Case-Based Planning. *Artificial Intelligence Review*, 16(1), ss. 3–36.

The Wargus Team (2002). *Wargus* (Version: 2.2.7) [Datorprogram]. The Wargus Team. http://wargus.sourceforge.net/index.shtml

World Chess Federation (2012). *FIDE Rating Regulations Effective From 1 July 2014*. http://www.fide.com/fide/handbook.html?id=172&view=article [2015-02-09]

World Chess Federation (u. å.). *Laws of Chess: For competitions Starting On or After 1 July 2014*. http://www.fide.com/fide/handbook.html?id=171&view=article [2015-02-09]

Appendix A - Designdokument etc.