Examensarbete

HÖGSKOLAN I SKÖVDE

EN SCHACK AI MED VARIERBAR SKICKLIGHETSNIVÅ BASERAD PÅ CASE-BASED REASONING

A CASE-BASED REASONING APPROACH TO A CHESS AI WITH A VARIABLE SKILL LEVEL

Examensarbete inom huvudområdet Datavetenskap
Grundnivå 30 högskolepoäng
Vårtermin 2015

Johannes Qvarford

Handledare: Peter Sjöberg
Examinator: Anders Dahlbom

Innehållsförteckning

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 deras motståndare 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). 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. **(Första delen av stycket känns mindre relevant nu när målet är att skapa en AI som använder CBR med justerbar skicklighetsnivå. Låta det vara för att det är någorlunda relevant, eller försöka hitta på något nytt så att problembeskrivningen känns mindre främmande?)**

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 sin svårighetsgrad. Svårighetsgraden ska överensstämma med spelarnas rank, så att agenten är svårare att besegra om dess fallbas är baserad på högt rankade spelare än om den är baserad på lågt rankade spelare. Fallbaser kommer kunna skapas baserade på tidigare spelade matcher dokumenterade i *Portable Game Notation* (PGN). Flera organisationer dokumenterar schackmatcher i detta format, som Universal Chess Federation (FIDE). Arbetet ska kunna användas som grund för att utveckla AI-agenter för spel med justerbar svårighetsgrad.

2 Bakgrund

I denna sektion presenteras bakgrundsinformation om ämnen och termer som nämns i arbetet. I sektion 2.1 presenteras CBR, besläktade termer och hur arbetet bygger på tidigare forskning kring CBR. Sektion 2.2 innehåller reglerna till schack, och termer som ofta används inom schack.

2.1 Case-based Reasoning

CBR är en teknik för problemlösning inom AI som är baserad på idén att använda lösningar på tidigare, liknande problem (Richter & Weber 2013). 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.

Nära besläktat med CBR är tekniken *Case-Based Planning* (CBP) som enligt Spalazzi (2001) 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 dem.

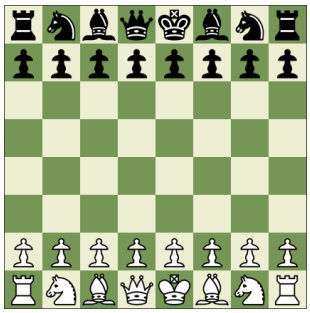
CBR och CBP har applicerats på datorspel i ett antal tidigare arbeten. Rekabdar, Shadger och Osareh (2012) presenterar hur CBP kan användas för att lära agenter att spela ett fotbollsspel tillsammans genom att låta dem observera hur ett motståndarlag spelar. I arbetet av Aha, Molineaux och Ponsen (2005) visas ett sätt att hantera den stora mängd fall som uppstår i *Real-Time Strategy* (RTS) spelet Wargus (The Wargus Team 2002) när CBP appliceras. 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.

Tidigare arbeten har använt mycket domänspecifik information inom respektive spel för att upptäcka planer som experterna utför implicit. Detta arbete lägger mindre fokus på att upptäcka djupare motivationer bakom experternas handlingar och utgår ifrån att fall i schack går att studera i isolation. Det lämnas till framtida arbeten att undersöka hur schackspecifik planplanering som öppningar och slutspel kan appliceras inom CBP. En aspekt som de ovannämnda arbetena delar är att de använder minst en expert per unikt beteende. Det är oklart om det går att producera ett antal beteenden från ett lägre antal experter och detta arbete bygger på att besvara denna oklarhet.

2.2 Schack

2.2.1 Regler

Schack är ett turbaserat brädspel för två spelare där målet är att besegra sin motståndare (FIDE 2014c). Spelet utspelar sig på en 8x8-rutors spelplan, där varje spelare kontrollerar varsin armé av spelpjäser – vit och svart. 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 betraktas det som att pjäsen hotar rutan, eller pjäsen som står på rutan.

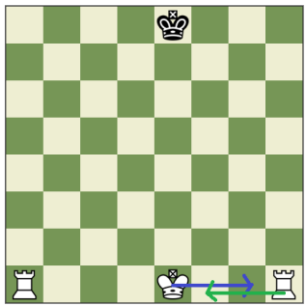
Bonden (♟) kan flytta sig ett steg rakt framåt (sett från den ägande spelarens håll), eller ett steg diagonalt framåt om draget är ett fångande drag. Springaren (♞) kan flytta sig två steg horisontellt eller vertikalt, och ett steg på den resterande axeln. Springaren kan flytta till en ruta även om det 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.

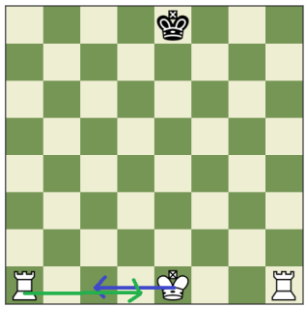
Bonden kan flytta två steg rakt framåt på sitt första drag. Om en spelare flyttar en bonde två steg, så kan bonden betraktas som om den bara tog ett steg, om den fångas av en motståndarbonde nästa drag. Detta kallas *en passant* och illustreras i Figur 2.

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. Detta kallas kort respektive lång rocked.

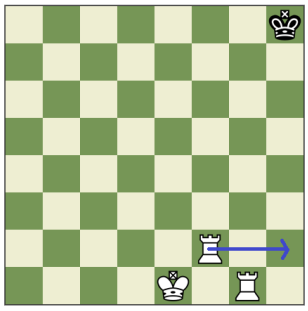


Figur 3 Bild som visar hur pjäserna flyttas när vit gör kort rockad.



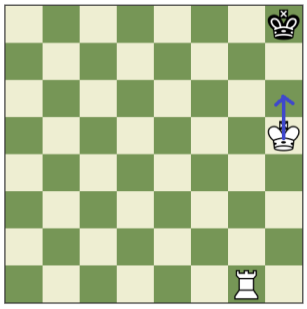
Figur 4 Bild som visar hur pjäserna flyttas när vit gör lång rockad.

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 remi. Ett exempel av remi visas i Figur 6.



Figur 6 Bild som visas hur vit kan göra remi. 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-rankning

Elo-rankning är ett sätt att ranka schackspelare relativt till varandra (FIDE 2014a). Enligt Elo-rankningssystemet 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.

2.2.3 Portable Game Notation

PGN är ett format som utvecklades för att spara och beskriva schackmatcher (”Standard: Portable Game Notation Specification and Implementation Guide” 1994). Ett PGN-dokument kan innehålla ett antal matcher och varje match innehåller metainformation om matchen och de drag som utfördes i matchen. Informationen kan gälla när eller var matchen spelades och av vilka. Dragen skrivs med algebraisk notation (AN).

(Citatet i första meningen i förra paragrafen är till en artikel utan känd författare. Hittade inte ett exempel på detta i det svenska systemet, så jag använda APA för citatformatet och referensen.)

AN är en notation som beskriver drag kortfattat till den grad att de inte är tvetydiga. Raderna numreras från vits synvinkel med bokstäver från a till h, och kolumnerna med siffrorna 1 till 8. En ruta på spelplanen kan då beskrivas med dess tillhörande rad och kolumn t.ex. e4 eller

a2. 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: e4, Nf3, Bb5. Om draget är ett fångande drag så sätts ett x framför rutan som pjäsen flyttades till. De drag som leder till schack har ett plustecken som suffix. Kort rockad representeras med ”O-O” och lång rockad 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.

2.2.4 Dagens schackdatorprogram

Ett datorprogram för att spela schack kräver två saker. Det första är ett användargränssnitt för en spelare att utföra drag och se vad dess motståndare gör för drag. Detta kan antingen vara ett konsolprogram där användaren använder text för att kommunicera med programmet, eller ett grafiskt användargränssnitt (GUI) där användaren interagerar med programmet genom att klicka och dra i bilder, knappar, fönster osv.. Det andra som schackdatorprogram kräver är ett antal AI-agenter som en spelare kan spela mot. Dessa AI-agenter kallas för schackmotorer. För att underlätta utvecklingen av olika schackmotorer har standardiserade kommunikationsprotokoll definierats, så att det inte krävs olika versioner av schackmotorer för olika användargränssnitt. XBoard (Mann & Muller 2009) är ett kommunikationsprotokoll mellan schackmotorer och användargränssnitt, baserat på gränssnittet till programmet med samma namn. När en version av spelet XBoard skapades för Windows-familjen av operativsystem, gavs det namnet WinBoard. *Universal Chess Interface* (UCI) (Rupert, 2006) är ett alternativ till XBoard och tillsammans används de av ett flertal användargränssnitt (Tey i.d.).

3 Problemformulering

3.1 Problembeskrivning

Syftet med det här arbetet är att undersöka hur CBR kan appliceras för att utveckla AI-agenter med varierbar svårighetsgrad. CBR har tidigare använts för att skapa AI-agenter som beter sig likt de experter de är baserade på. Mycket av denna forskning har dock använt en expert per beteende. Detta innebär att det krävs minst lika många experter som antalet beteenden AI-agenterna ska ha. Det finns lite forskning huruvida insamlad data från experter av olika skicklighetsnivåer kan kombineras och producera beteenden vars skicklighet ligger mellan experternas nivåer. Om så är fallet skulle CBR kunna användas för att skapa AI-agenter som har fler skicklighetsnivåer än antalet experter den är baserad på. Detta skulle minska resurserna som skulle krävas för utvecklare att samla in expertdata, samtidigt som spelare skulle erbjudas ett större urval av skicklighetsnivåer när de spelar mot AI-agenter.

I detta arbete används spelet schack som exempel för att undersöka hur väl tekniken går att utföra. För att vara av intresse för datorspelutveckling måste lösningen även kunna implementeras realistiskt med de resursbegränsningar som datorspel förväntas ha. AI-agenten måste kunna köras på vanlig konsumenthårdvara och utföra sina drag inom rimlig tid. I FIDE-tävlingar får en schackspelare 90 minuter på sig att utföra sina första 40 drag (2014b), vilket är det krav som AI-agenten förväntas följa. AI-agenten behöver inte kunna spela lika bra som någon annan schackmotor eller spelare, utan det viktigaste AI-agenten ska visa är hur blandningar av expertdata kan producera olika beteenden.

3.2 Metodbeskrivning

För att AI-agenten ska kunna spela schack måste den ha en fallbas baserad på expertdata. AI-agenten kommer använda expertdata i form av PGN filer av tidigare spelade matcher. Matcherna har spelats av flera olika spelare med olika ELO-rankning. När AI-agenten ombeds att spela på en given skicklighetsnivå, ska den leta upp matcher av spelare vars ELO-rankning bäst matchar den efterfrågade skicklighetsnivån och skapa en fallbas av dessa. Om det inte finns någon spelare vars rakning matchar den efterfrågade skicklighetsnivån, ska matcher av spelare med både högre och lägre rankning användas för att skapa fallbasen. När AI-agenten ombeds göra ett drag ska den konsultera fallbasen och utav de fall vars drag går att utföra i matchens nuvarande läge, ska den utföra draget i det fall vars läge är mest likt det läge som matchen befinner sig i.

AI-agenten ska implementeras som ett schackmotordatorprogram. Anledningen till att ett datorprogram används för att visa AI-agentens beteende är att det kan vara svårt att förutsäga hur AI-agenten kommer bete sig givet den stora mängden expertdata som måste undersökas. Programmet ska följa kommunikationsprotokollet UCI. Xboard har övervägts, men har valts bort på grund av att det verkar svårare att implementera. **(Egen tanke, kan inte styrka med pålitlig källa. Ta bort/ha kvar?)**

Varje spelare som ska bidra med expertdata ska ha en skicklighetsnivå som överensstämmer med deras rankning. Utöver dessa skicklighetsnivåer, ska det finnas ytterligare ett antal nivåer som distribueras jämt mellan spelarnas skicklighetsnivåer. För att se hur bra AI-agenten spelar på olika skicklighetsnivåer ska AI-agenten spela ett antal matcher mot sig själv, med alla kombinationer av svårighetsgrader. Datorprogrammet Arena (Blume 2014) ska användas

för att utföra matcherna. Resultatet kan visa om AI-agentens presterar som förväntat givet dess skicklighetsnivå.

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.

Blume, M. (2014). *Arena* (Version: 3.5) [Datorprogram]. Blume, M. http://www.playwitharena.com/

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.)

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.

Standard: Portable Game Notation Specification and Implementation Guide (1994). http://www6.chessclub.com/help/PGN-spec [2015-02-16]

Tey, A. (i. d.). *Interface Support for Winboard/UCI*. http://horizonchess.com/FAQ/Winboard/interface.html [2015-02-16]

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

World Chess Federation (2014a). *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 (2014b). *General Rules and Recommendations for Tournaments: Time Control*. www.fide.com/component/handbook/?id=39&view=category [2015-02-09]

World Chess Federation (2014c). *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