Gruppenunterricht zum Thema Spieltheorie

Autoren:
Martin Hirt, Daniel Matter, Rolf Bänziger, Werner Hartmann

Überarbeitet von Tobias Schlatter
Fassung vom März 1999

Inhalt

Einführung

Bei der „Spieltheorie“ geht es um die Frage: Wie können wir ein Spiel wie **Schach** oder **Vier Gewinnt** auf einem Computer programmieren?
In der beiliegende Einführung in die Spieltheorie erhalten wir das nötige Grundwissen. Die Bearbeitung dauert etwa zwei Lektionen.
In den folgenden Lektionen wollen wir auf eine etwas andere Art und Weise verschiedene Gebiete der Spieltheorie kennenlernen. Diese Form des Unterrichts wird "Puzzle" genannt. Zuerst kriegt jeder Schüler ein Thema, das er selbständig vertieft. Er wird also Experte in diesem Gebiet. Anschliessend setzen sich die Experten jedes Gebiets zu einer Expertengruppe zusammen und diskutieren, wie sie ihr Stoffgebiet ihren Mitschülern vermitteln können. In einem dritten Teil, der sogenannten Unterrichtsrunde, bilden wir Gruppen, in denen jedes Fachgebiet von einem (in Ausnahmefällen von zwei) Experten vertreten wird. Jeder Experte unterrichtet sein Fachgebiet den Mitschülern.

Die Themen:

Gruppe 1: **Alpha-Beta-Pruning**
Wie kann der Minimax-Algorithmus beschleunigt werden? Hier lernen wir eine Methode kennen, wie wir die Suche nach dem besten Zug um ein Vielfaches beschleunigen können. Und das Besondere: Wir finden immer den gleichen Zug wie bei einem normalen Minimax-Algorithmus. Also keine Reduktion der Spielstärke!

Gruppe 2: **Vier interessante Begriffe in der Spieltheorie**
Hier lernen wir einige Stärken und Schwächen des Minimax-Algorithmus kennen. Insbesondere beschäftigen wir uns damit, welche Auswirkungen die Konstanz der Suchtiefe hat. Ausserdem in diesem Kapitel: Warum können wir **Vier Gewinnt** nicht vollständig analysieren?

Gruppe 3: **Heuristik**
Was ist Heuristik? Wie können wir eine Spielstellung exakt bewerten? Wir gehen der Frage nach, wie wir etwas herausfinden können, ohne Methoden zur Verfügung zu haben. Ausserdem: Eine mögliche Beschleunigung des Minimax-Algorithmus, die allerdings eine geringere Spielstärke verursacht.

Gruppe 4: **Computerlernen**
Was gibt es in der Spieltheorie ausser Minimax? Wie können wir ein Spiel programmieren, ohne dass wir uns auch nur einen Gedanken über Spielstrategien machen müssen? Kann ein Computer lernen? Oder: Ist ein Computer sogar intelligent?
Solche und ähnliche Fragen wollen wir uns in diesem Teil stellen. Ein kurzer Ausflug in die künstliche Intelligenz rundet das Ganze ab.

Einführung in die Spieltheorie

Vielleicht haben Sie sich auch schon gefragt, wie überhaupt ein Spielcomputer funktioniert. Wie „merkt“ der Computer, was ein guter Zug ist und wie findet er den besten Zug? Warum spielt ein Computer besser als ein anderer? Und wenn die Computer immer schneller werden, warum verlieren sie trotzdem gegen die besten menschlichen Spieler? Im Laufe der folgenden Lektionen zum Thema Spieltheorie werden Sie Antworten auf einige von diesen Fragen bekommen.
Falls Sie sich schon ein bisschen für die Spieltheorie interessieren, dann befinden Sie sich in guter Gesellschaft: Im folgenden Bericht aus dem Internet sehen Sie ein bisschen in die Welt der Wissenschaftler hinein, die sich mit der Spieltheorie ganz praktisch auseinandersetzen. Der Artikel ist im Internet unter http://www.zdnet.de/news/artikel/1997/05/12004-wf.htm auf den Seiten des Ziff-David-Verlags zu finden.

Sensation: Schach-Rechner schlägt Grossmeister

tk - Der Supercomputer Deep Blue von IBM (Branchenspitzname: Big Blue) schlägt den Schach-Grossmeister Garry Kasparov. Damit gewinnt erstmals ein Computer nicht nur über einen amtierenden Weltmeister. Experten halten Kasparov für den derzeit besten lebenden Schachspieler überhaupt. Hatte die Schach-Ikone noch im letzten Jahr die Auseinandersetzung gegen die Maschine gewonnen, musste sich der Russe nun geschlagen geben.
Das über sechs Partien gehende Match (dauerte eine Woche; komplette Dokumentation unter www.chess.ibm.com/home/b.html) endete 3,5 zu 2,5 für Deep Blue. Zwei der Partien gewann der auf Schach getrimmte Super-Rechner, drei endeten Unentschieden und nur eine Partie verlor Deep Blue. Die letzte und entscheidende Partie eröffnete Kasparov mit einer Caro-Kann-Verteidigung. Zwar verlor Deep Blue dadurch einen Springer, trotzdem war der Schach-Grossmeister mit dieser Strategie chancenlos. Nach Verlust eines eigenen Springers, Läufers, Bauers und der Königin gab Kasparov auf. Die Spieldauer: Knapp eine Stunde oder 19 Züge.
"Ich muss mich für die heutige Leistung entschuldigen", so der 34jährige Kasparov in der Pressekonferenz. "Ich hatte keine richtige Energie für die Auseinandersetzung." Laut Beobachtern hatte der IBM-Rechner mit den ersten fünf Partien (bis dahin eine Niederlage, ein Sieg, drei Unentschieden) Kasparov's unerschütterliches Selbstbewusstsein der beste Schachspieler aller Zeiten zu sein, erschüttert. So gab Kasparov nach der, mit einem weiteren Unentschieden endenden fünften Partie auch zu: "Ich muss mich vorsehen, denn ich kann jeden Spieler der Welt ausrechnen, aber ich kann nicht diese Maschine ausrechnen."
Deep Blue ist ein Computer der IBM-RS/6000-SP-Reihe. Seine Parallel-Prozessor-Architektur (Taktrate von 130 Megahertz) mit für Schach optimierten Mikrochips kann zwei Millionen Positionen pro Sekunde errechnen. Verlor Deep Blue noch im letzten Jahr die erste Auseinandersetzung mit Kasparov (damals gewann der Grossmeister drei Partien, verlor eine und zwei endeten unentschieden), verdoppelten IBM-Techniker in den letzten Monaten die Rechenleistung des Computers auf den heutigen Stand. Durch den Sieg strich das IBM-Team ein Preisgeld in Höhe von 700.000 Dollar ein, Kasparov muss sich mit 400.000 Dollar zufrieden geben. Mit Deep Blue will IBM die Entwicklung von Super-Rechnern für komplexe und umfangreiche Berechnungen vorantreiben. Bisherige Anwendungen sind die Wettervorhersage,

die Arzneimittel-Forschung und die Kontrolle des Flugverkehrs. Folgend die Daten der Schachgegner.

	Garry Kasparov	Deep Blue
Gewicht:	79,8 Kilogramm	1,4 Tonnen
Alter:	34 Jahre	4 Jahre
Geburtsort:	Aserbaidschan	USA
Prozessoren:	50 Milliarden	32 Risc Chips
Positionen/Sekunde:	3 - 5	2 Millionen
Energiequelle:	elektro-chemisch	elektrisch
Bestimmung:	Schachweltmeister	Wettervorhersage

1. Zwei-Personen-Spiele, Spiele mit perfekter Information

Die meisten Spiele, die man gegen den Computer spielen kann, sind Zwei-Personen-Spiele. Typische Beispiele dafür sind Spiele wie SCHACH, GO, VIER GEWINNT oder OTHELLO. Meistens haben diese Spiele aber noch eine zweite wichtige Eigenschaft: Jeder Spieler hat alle Informationen über den aktuellen Spielstand. Beide wissen also von den Spielfiguren ihres Gegners alles. Ein solches Spiel nennt man ein Spiel mit perfekter Information. Nicht zu dieser Kategorie gehört zum Beispiel der JASS. Ich weiss zwar, wie viele Karten mein Gegenüber hat, und wenn ich ein guter Spieler bin, habe ich mir auch seine schon gespielten Karten gemerkt. Aber im Normalfall kann ich nicht wissen, welche Karten er noch in der Hand hält.
Wir werden uns im Folgenden nur mit Zwei-Personen-Spielen mit perfekter Information beschäftigen.

2. Der Spielbaum

In einem solchen Spiel wird abwechslungsweise gezogen. Bei jedem Zug gibt es verschiedene Möglichkeiten, zu spielen. Bei einem VIER GEWINNT z.B. hat es sieben Schlitze, in die Spielsteine geworfen werden können (siehe Bild 1).
Beim Spielbeginn ist das Gitter leer. Derjenige, der das Spiel eröffnet, kann seinen Stein in eine beliebige der sieben Kolonnen werfen. Er hat also sieben verschiedene Zugs-Möglichkeiten. Danach hat auch sein Gegner bei seinem ersten Zug noch sieben Varianten offen. Im weiteren Spielverlauf kann es jedoch vorkommen, dass in einen Schlitz kein Stein mehr geworfen werden kann, weil diese Kolonne schon voll ist. Dies ist im Bild 1 z.B. in der dritten und vierten Kolonne der Fall. Darum hat der Spieler mit den schwarzen Steinen nur noch fünf Möglichkeiten zur Auswahl.

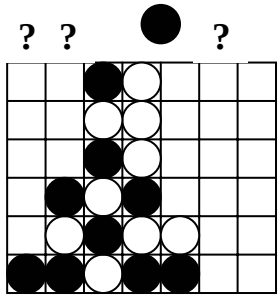


Bild 1 - Vier Gewinnt

Jede beliebige Spielsituation ist ja eigentlich nur ein momentaner Zustand. Durch irgendeine Zug-Reihenfolge ist der aktuelle Spielstand entstanden und mit einem weiteren Zug wird dann der Zustand wieder verändert. Einen solchen Zustand wollen wir als Knoten bezeichnen. Von einem Zustand in den nächsten kommt man, indem man einen Spielzug

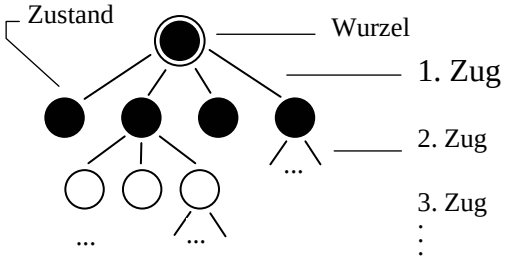


Bild 2 - Spielbaum

ausführt. Einen Zug verbindet also zwei Zustände. Deshalb wird ein Zug als Kante dargestellt. Man kann nun alle möglichen Spielzüge eines VIER GEWINNT aufzeichnen (Bild 2): Der Anfangsknoten ist der Zustand, bei dem alle Kolonnen noch leer sind. Dieser Knoten heisst in der Fachsprache Wurzel. Wie wir oben gesehen haben, gibt es nun sieben Möglichkeiten, wie der beginnende Spieler seinen ersten Stein setzen kann. Diese Möglichkeiten werden durch die sieben Kanten dargestellt, die von der Wurzel weggehen. Jede dieser Kanten führt zu einem Knoten in der zweiten Ebene. Diese Knoten sind also alle möglichen Zustände nach dem ersten Zug. Von jedem dieser Knoten aus geht das Spiel natürlich wieder weiter und für jeden möglichen Zug des zweiten Spielers ist wieder eine Kante eingezeichnet.
Das ganze Gebilde sieht jetzt wie ein auf den Kopf gestellter Baum aus. Und weil dieser Baum alle Spielmöglichkeiten darstellt, wird er Spielbaum genannt. Einen solchen Spielbaum gibt es aber nicht nur für das VIER GEWINNT, sondern für jedes beliebige Zwei-Personen-Spiel mit perfekter Information.
Wenn man nun einen frei wählbaren Weg entlang der Kanten nach unten geht, kommt man irgendwann zu einem Knoten, von dem keine Kanten mehr weitergehen. Einen solchen Endknoten bezeichnet man als Blatt. Ein Blatt im Baum entspricht also einem Zustand, nachdem nicht mehr weitergespielt werden kann und demzufolge das Spiel zu Ende ist. Es gibt drei verschiedene Arten solcher Endzustände (Bild 3):

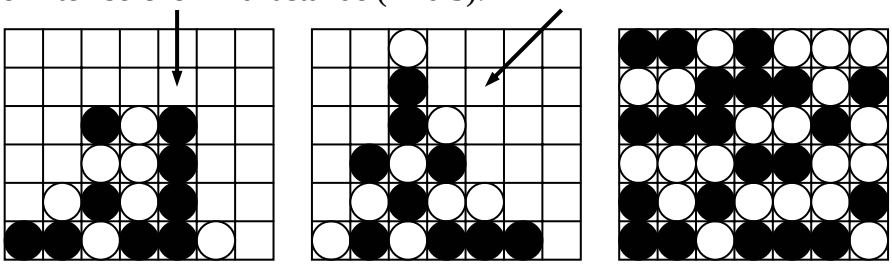


Bild 3 – Verschiedene Spielendzustände

3. MAX- und MIN-Spieler

In einem Spiel gibt es ja normalerweise 'gute' und 'schlechte' Züge. Nach einem guten Zug ist meine Situation vorteilhafter als vorher, durch einen schlechten stehe ich weniger gut da. Ebenso gibt es Züge des Gegners, die meine Stellung mehr oder weniger fest verschlechtern. Wenn ich an der Reihe bin, versuche ich natürlich, einen möglichst guten Zug zu machen. Umgekehrt wird mein Gegenspieler seinerseits optimal für sich spielen wollen.
Den Spieler, den wir jetzt anschauen werden, ist ein sehr guter, aber auch ein vorsichtiger und pessimistischer Spieler. Wir wollen ihn MAX nennen. Wenn der MAX am Zug ist, spielt er den bestmöglichen Zug. Er ist bestrebt, den maximalen Zug herauszufinden. Aus diesem Grund wurde er auch MAX genannt.
Sein Gegner heisst MIN. Auch dieser wird optimal spielen. Das bedeutet für Max natürlich, dass dies der ungünstigste (oder minimale) Zug sein wird. Darum wurde sein Gegenspieler auch MIN genannt.
In der Praxis ist es natürlich schwierig festzustellen, welches nun wirklich der beste Zug ist. Ein Zug kann im Moment ungünstig aussehen, aber einige Spielzüge später kann sich die Situation ganz gewendet haben und der Spieler gewinnt. Es kann aber auch vorkommen, dass ein vermeintlich guter Zug zur Katastrophe führt.
Wir wollen nun anhand eines einfachen Beispiels (Bild 4) herausfinden, wie man den besten Zug finden kann:
Die Wurzel des Spielbaumes stellt die aktuelle Spielsituation dar. Am Anfang ist MAX am Zug. Er spielt mit den schwarzen Steinen. Auf dem Bild sind alle weiteren Spielzüge eingezeichnet, die noch möglich sind. Die Blätter im Spielbaum sind mit 1, ½ oder 0 bezeichnet,

abhängig davon, ob MAX gewinnt, unentschieden spielt oder verliert. Von diesen Spiel-Endständen aus kann man nun den Baum rückwärts analysieren, um herauszufinden, welches der beste Zug für MAX ist. Wie man dabei vorgeht, wollen wir im nächsten Abschnitt anschauen.

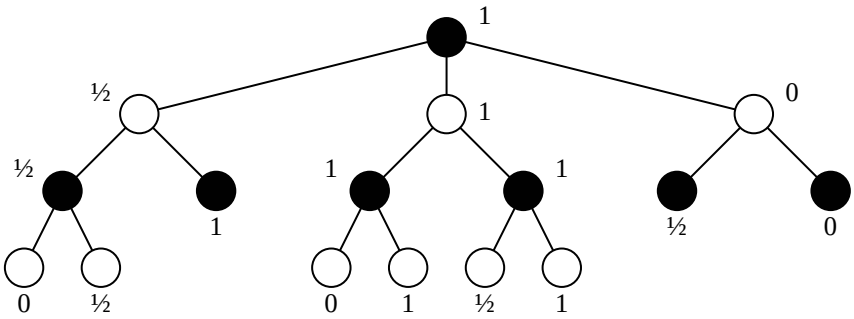


Bild 4 – MIN-MAX-Baum

Vorgehen zur Bewertung der Knoten:
Wir beginnen links unten bei den Endständen 0 und ½: Der letzte Spielzug wurde von MAX getätigt. Da dieser immer den besten Zug wählt, wird er in diesem Fall auf ½ spielen. Deshalb bezeichnen wir den schwarzen MAX-Knoten mit ½.
Nun gehen wir eine Stufe höher zum weissen Knoten. Für MIN stehen die Möglichkeiten ½ oder 1 offen. Da er bei 1 verliert, wird er natürlich ½ (Unentschieden) wählen.
Um nun die Wurzel bewerten zu können, müssen zuerst die andern beiden Äste ausgewertet werden. Beim mittleren Ast ist das Ergebnis 1, beim rechten 0. Der MAX-Spieler wird also für seinen nächsten Zug den mittleren Ast wählen, da dieser am höchsten bewertet wurde. Über diesen Ast gelangt er dann schlussendlich auch nach drei Zügen zum Sieg.

4. Die Suchtiefe

Ein solcher Spielbaum ist für die ersten drei bis vier Züge noch einigermassen darstellbar. Will man ihn aber noch weiter aufzeichnen, so wächst der Baum sehr schnell auf unüberblickbare Grösse an. Beim VIER GEWINNT hat er schon nach sieben Zügen mit je sieben Zugs-Möglichkeiten schon beinahe 1 Million Blätter! Es ist deshalb unmöglich, auch nur ein kleineres Spiel wie das VIER GEWINNT vollständig zu analysieren. Bei einem komplizierten Spiel wie SCHACH steigt der Aufwand schon beinahe ins Unermessliche! Selbst mit schnellsten Computern stösst man rasch an Grenzen. Die besten Schachcomputer berechnen den Spielbaum heute (1993) etwa für die nächsten 10 Spielzüge.
Vom ganzen Baum werden also nur die Knoten der ersten 10 Ebenen berechnet. Um den besten Zug zu suchen, gehe ich also 10 Stufen tief den Baum hinunter. Deshalb nennt man die Anzahl dieser berechneten Stufen die Suchtiefe. Wenn der Computer beim Suchen des besten Zuges auf der Ebene der Suchtiefe angelangt ist, berechnet er keine weiteren Züge mehr. Deshalb wird hier von der Suchgrenze gesprochen.
Es ist natürlich ideal, wenn die Suchtiefe so gross ist, dass man bei der Suche zu Blättern gelangt, weil dort der Spielstand klar ist. Normalerweise ist dies leider nicht der Fall. Es stellt sich deshalb die Frage, wie die Knoten an der Suchgrenze bewertet werden sollen, wenn der Spielstand nicht eindeutig ein Gewinn von Schwarz oder Weiss ist. In einem solchen Fall muss deshalb die aktuelle Spielsituation bewertet werden. Diese Bewertung heisst im Spieltheorie-Jargon Feldbewertung.

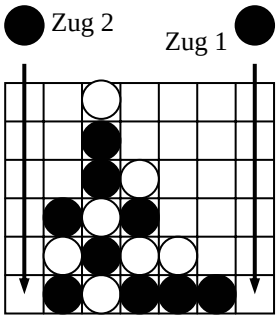


Bild 5 – Zugauswahl

Beispiel (Bild 5): Schwarz ist am Zug.
Schwarz spielt Zug 1. Schwarz gewinnt. Die Stellung hat also den maximalen Wert für Schwarz. Die Feldbewertung der Stellung nach diesem Zug gibt deshalb auch den maximalen Wert zurück. (z.B. 1, wenn 1 das Maximum ist).
Schwarz spielt weder Zug 1 noch Zug 2. Weiss wird nachher natürlich Zug 2 spielen und somit seinerseits gewinnen. Dann ist der Wert der Spielstellung von Schwarz aus gesehen minimal (z.B. -1, wenn -1 das Minimum ist), weil er verloren hat.
Schwarz spielt Zug 2. Die Situation ist wieder recht ausgeglichen. Die Feldbewertung wird also einen Wert um Null herum ergeben. Einen genauen Wert erhält man, wenn man die Spielkonstellation nach bestimmten Kriterien untersucht. Solche Kriterien wären z.B. Zugzwang (Weiss muss nun ganz rechts spielen, um nicht zu verlieren) oder die Anzahl der Reihen mit drei Steinen, die noch zu einer 4er-Reihe aufgefüllt werden können (Schwarz hat deren zwei, Weiss nur eine). Dies ist nur eine kleine Auswahl von Kriterien. Man könnte noch viele andere herausfinden und anfügen.

5. Die Minimax-Regel

Folgende Regeln beschreibt das Vorgehen, wie der Wert eines beliebigen Knotens im Baum berechnet werden kann:

1. Der Knoten ist ein MAX-Knoten: Der Wert ist das Maximum von allen direkten Nachfolgern.
2. Der Knoten ist ein MIN-Knoten: Der Wert ist das Minimum von allen direkten Nachfolgern.
3. Der Wert eines Knotens an der Suchgrenze (=Blatt) erhält man durch die Feldbewertung dieser Stellung.

Verständnis-Test

Sie haben in diesem Kapitel einen Überblick über die Spieltheorie bekommen und die Grundlagen für die weiteren Themen kennengelernt. Mit diesem Test können Sie sich nun selber prüfen, ob Sie die wichtigsten Begriffe verstanden haben. Er soll Ihnen aufzeigen, wo Sie noch Lücken haben, aber gleichzeitig können Sie damit das Gelesene vertiefen.
Der Test wird nicht benotet. Versuchen Sie ihn ohne die Unterlagen zu lösen. Nur so finden Sie heraus, ob Sie die einzelnen Punkte wirklich begriffen haben.

Aufgabe 1
Warum kann für ein Spiel ohne perfekte Information kein Spielbaum aufgestellt werden?

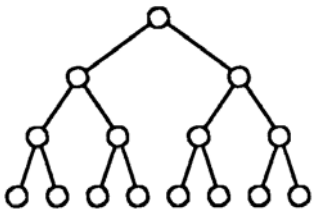
Aufgabe 2
Unten sind einige Bestandteile eines Spielbaumes aufgelistet. Beschreiben Sie mit wenigen Worten, was diese im wirklichen Spiel darstellen.

a) Wurzel _____ c) Kante _____

b) Knoten _____ d) Blatt _____

Aufgabe 3
Welches ist die wichtigste Eigenschaft des MAX-Spielers?

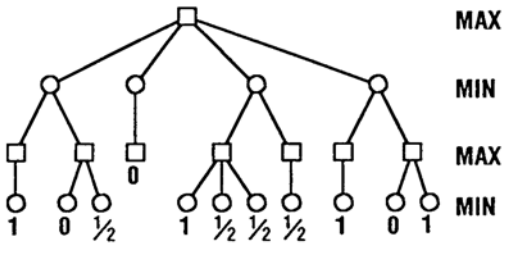
Aufgabe 4
Wie gross ist die Suchtiefe dieses Baumes? _____



Aufgabe 5
Nun haben Sie viele theoretische Fragen beantwortet. Da aber ein Spiel eher praktisch ist, sollen Sie zum Schluss noch eine praktische Aufgabe lösen:

Sie haben untenstehend einen Spielbaum mit den Endzuständen 1, 0, ½ (MAX gewinnt, verliert, spielt unentschieden). Finden Sie den besten Zug für Max heraus!

Tip: Minimax-Regel. Versuchen Sie die Aufgabe zuerst ohne die Unterlagen zu lösen. Falls Sie nicht durchkommen, können Sie die Regeln zu Hilfe nehmen.

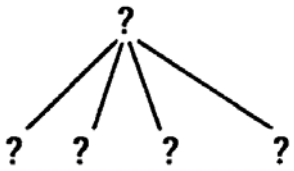


Lösungen zum Verständnis-Test

Vergleichen Sie Ihre Antworten mit den untenstehenden Lösungen. Die Aufgaben 1-3 sollten in den wichtigsten Zügen mit den Lösungen übereinstimmen. Falls eine Ihrer Antworten falsch oder nur teilweise richtig ist, müssen Sie den entsprechenden Abschnitt in Ihren Unterlagen nochmals genauer studieren. Falls dann immer noch Unklarheiten bestehen, dürfen Sie mit Ihrem Nachbarn darüber diskutieren. Wenn Ihnen auch dieser nicht weiterhelfen kann, dann fragen Sie den Lehrer.

Aufgabe 1

Bei einem Spiel ohne perfekte Information fehlen wichtige Informationen wie z.B. die aktuelle Spielstellung. Es können deshalb keine Folgezüge und somit auch kein Baum konstruiert werden. (siehe Bild)



Aufgabe 2

a) Wurzel = Aktuelle Spielstellung
b) Knoten = Spielzustand
c) Kante = Spielzug
d) Blatt = Endstand, Spiel ist zu Ende

Aufgabe 3

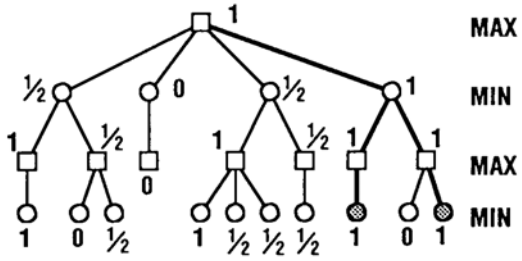
Der MAX-Spieler sucht immer den besten (=maximalen) Zug.

Aufgabe 4

Die Suchtiefe ist drei, denn der Computerspieler schaut drei Züge voraus.

Aufgabe 5

Der MAX-Spieler wird den Zug ganz rechts spielen. Er kann dort gewinnen, egal was MIN nachher spielt.



Die α-β-pruning Methode

Übersicht

Sie haben in der Einführung den Spielbaum (oder Minimax-Baum) kennengelernt. Dieser Baum hat die negative Eigenschaft, dass er sehr schnell sehr riesig wird, weil die Anzahl möglicher Züge mit der Suchtiefe exponentiell steigt. In diesem Teil lernen Sie eine Technik kennen, die es ermöglicht, den besten Zug eines Spielbaumes zu berechnen, ohne alle Züge durchprobieren zu müssen.

Vorgehen

1. Studieren Sie den Lehrtext "Die α-β-pruning Methode". (Zeitaufwand: ca. 45 Min.)
2. Lösen Sie nun die Aufgaben der "Schüler-Lernkontrolle". (Zeitaufwand: ca. 15 Min.)
3. Besprechen Sie mit den andern Experten Ihres Themas, wie ihr eure Mitschüler unterrichten wollt. (Zeitaufwand: ca. 30 Min.
4. Falls Sie zusätzliche Vorbereitungen für die Unterrichtsrunde brauchen, erledigen Sie diese als Hausaufgabe.

Lernziele für die ganze Klasse

1. Sie können erklären, warum mit der α-β-Methode nicht alle Knoten eines Spielbaumes berechnet werden müssen, wenn der Minimax-Wert der Wurzel berechnet werden soll.
2. Sie können den Wert des Wurzelknotens eines einfachen Spielbaumes mit Hilfe der α-β-Regeln berechnen.
3. Sie wissen den Unterschied zwischen der α- und der β-Grenze (α-bound / β-bound).

Die α-β-pruning Methode

1. Einleitung

In der Einführung in die Spieltheorie haben Sie den Begriff des Spielbaumes kennengelernt. Dieser besteht aus einer Wurzel und Ästen, die sich von der Wurzel aus verzweigen. Dabei stellt die Wurzel den aktuellen Spielstand dar, und die Äste sind die möglichen Spielzüge (Bild 1). Das Ziel ist ja nun, vom momentanen Spielstand aus möglichst weit voraus zu blicken, um den bestmöglichen Zug zu spielen. Dazu müssen alle möglichen Spielzüge herausgefunden und bewertet werden, d.h. der Spielbaum muss möglichst tief analysiert werden. Je weiter aber der Baum berechnet wird, desto grösser wird bekanntlich auch der Aufwand. Da dieser sehr schnell ansteigt, suchte man schon bald nach Auswertungstechniken, die einem erlauben, mit geringerem Aufwand trotzdem alle Informationen zu erhalten. Eine dieser Techniken heisst α-β Pruning Procedure. Wir wollen uns nun genauer mit dieser auseinandersetzen.

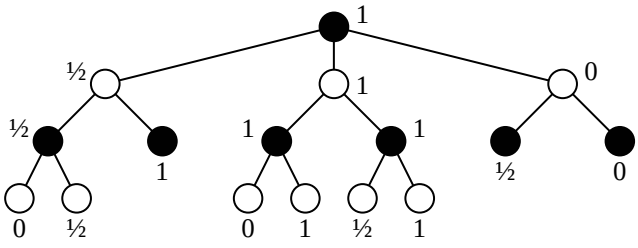


Bild 1 – Spielbaum

2. Beschreibung der α-β Methode

Der α-β Algorithmus ist die weitaus am häufigsten verwendete Technik, um die Suche nach dem besten Spielzug zu verschnellern ohne Information zu verlieren. Der Algorithmus bestimmt den Minimax-Wert der Wurzel. Im Beispiel Bild 1 ist dieser Wert eins. Den Baum wird nach einem vorher bestimmten Schema traversiert, z.B. von links nach rechts. Dabei werden alle Knoten ausgelassen, die den Wert der Wurzel nicht mehr beeinflussen können. Diese Methode ist im Bild 2 demonstriert, das einen Minimax-Baum mit der Suchtiefe vier zeigt:

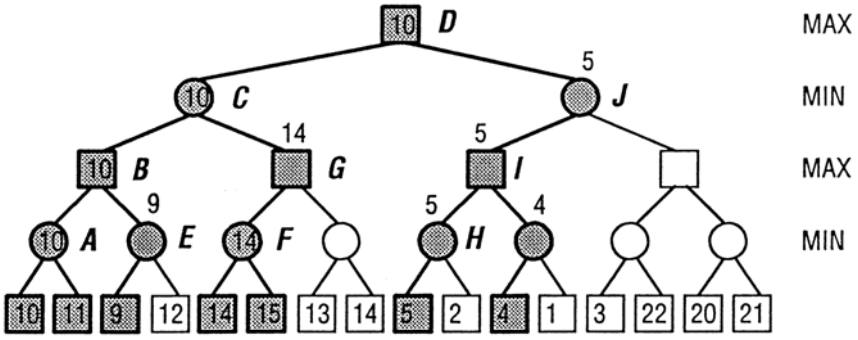


Bild 2 – Beispiel einer Spielbaum-Auswertung mit α-β Pruning

Die Zahlen innerhalb der Knoten der untersten Ebene zeigen die Ergebnisse der Feldbewertung von diesen Spielstellungen.
Die Zahlen auf den höheren Ebenen sind die Werte, die durch die α-β-Methode berechnet wurden. Wenn eine Zahl innerhalb eines Knotens geschrieben ist, dann entspricht sie dem tatsächlichen Wert dieses Knotens. Steht sie aber ausserhalb, dann heisst das, dass sich der Wert zwar nach Berechnung von weiteren Blättern noch verändern könnte, aber dass dies für den Minimax-Wert der Wurzel keine Auswirkung mehr hätte.
Die dicken Linien bezeichnen diejenigen Spielzüge, die durch die α-β-Procedure wirklich evaluiert wurden. Die dünnen Linien stehen für alle andern möglichen aber nicht berechneten Spielzüge.

Vorgehen:

1. Die α-βpruning Procedure beginnt von links. Als erstes werden also die Werte 10 und 11 berechnet. Da ihr Vorgänger A ein MIN-Knoten ist, hat dieser den Wert 10. (10 ist das Minimum von 10 und 11).
2. Um den Vaterknoten B von A zu berechnen, muss natürlich zuerst der Wert von dessem zweiten Sohn E bekannt sein. Der erste Nachfolger von E hat den Wert 9. Da E ein MIN-Knoten ist, bedeutet dies, dass sein Wert unabhängig von seinen anderen Nachfolgern nicht mehr grösser als 9 werden kann. Sein Vorgänger B verlangt aber einen möglichst grossen Wert. Da aber der Wert von A (10) grösser ist als der bestmögliche Wert von E, müssen die weiteren Nachfolger von E nicht mehr berechnet werden. B hat also sicher den Wert 10.
3. Um den Knoten C zu berechnen, brauchen wir die Information von seinem rechten Ast. Dazu müssen wir zuerst Werte an der Suchgrenze berechnen. Die ersten beiden Werte sind 14 und 15. Der MIN-Knoten F bekommt also den Wert 14 zugewiesen. Weil sein Vorgänger G ein MAX-Knoten ist, wird dieser also mindestens den Wert 14 haben. Da C aber das Minimum aller seiner Nachfolgerwerte nimmt und er schon einen Nachfolger mit Wert 10 hat, ist der rechte Ast mit einem Mindestwert von 14 nicht mehr von Bedeutung.
4. Der eine Nachfolger von D hat nun den Wert 10. Um einen besseren Zug als denjenigen in den Zustand C zu finden, müsste also der rechte Nachfolger von D ein höheres Resultat liefern als C. Um herauszufinden, ob er das kann, muss wiederum ein Knoten in der unter-

sten Ebene evaluiert werden. Der Knoten ganz links hat den Wert 5. Sein Vorgänger H - ein MIN-Knoten - kann darum höchstens den Wert 5 haben. Weil auch durch die weiteren Nachfolger der Wert von H nicht mehr grösser als 10 werden kann, müssen diese deshalb nicht mehr berechnet werden. Da der Knoten I das Maximum seiner Nachfolger nimmt, wäre es nun möglich, dass sein rechter Ast ein höheres Resultat als 5 ergibt. Doch der erste Knoten in der untersten Ebene hat den Wert 4, was bedeutet, dass der rechte Sohn von I nicht über diesen Wert hinauskommen kann. Da I also nicht grösser als 5 ist, kann auch sein Vaterknoten J höchstens den Wert 5 bekommen, da dieser den kleinstmöglichen Wert seiner Nachfolger annimmt. D hat also den Endwert 10.

Wir haben gesehen, dass man mit der α-β Methode unnötige Berechnungen weglassen kann. Im Spielbaum des Beispieles hat es sechzehn Spielstellungen auf der untersten Ebene. Bei einer vollständigen Suche müsste man deshalb sechzehnmal eine Feldbewertung durchführen. Von diesen sechzehn Stellungen mussten dank α-β pruning aber nur deren sieben ausgewertet werden. In der Praxis ist ein Spielbaum viel grösser, weil im Normalfall mehr als nur gerade zwei Züge zur Auswahl stehen. Wenn dann ein Ast nicht mehr evaluiert werden muss, kann man sehr viele Berechnungen einsparen. Bei Spielprogrammen wird deshalb immer mit der α-β-Methode gearbeitet.

3. Was bedeutet α-β-pruning?

Das englische Wort pruning wird übersetzt mit abschneiden von unnützen Teilen. Im Bereich der Landwirtschaft wird damit der Abschneide-Vorgang von toten oder lebenden Teilen einer Pflanze bezeichnet, der dazu dient, die Fruchtbarkeit zu erhöhen.
Im obigen Beispiel haben wir gesehen, dass unter bestimmten Bedingungen Teile des Spielbaumes nicht mehr evaluiert werden müssen. Man kann diese Bedingungen auf zwei Regeln reduzieren! Diese Regeln geben Grenzen an, die besagen, ab wo die Weitersuche nicht mehr nötig ist. Diese beiden Grenzen werden α- und β-Grenzen genannt, auf englisch α-bound und β-bound. Als Nächstes folgen nun die zwei Regeln, die wir anschliessend anhand von zwei Beispiele betrachten wollen:

Regel 1: α-bound
Die Abschneide-Grenze für einen MIN-Knoten wird α genannt. α ist gleich dem grössten Wert aller MAX-Vorfahren des MIN-Knotens. Sobald der aktuelle Wert des MIN-Knotens kleiner gleich α wird, kann die Auswertung abgebrochen werden. Deshalb ist α eine untere Grenze.

Regel 2: β-bound
Die Abschneide-Grenze für einen MAX-Knoten wird β genannt. β ist gleich dem kleinsten Wert aller MIN-Vorfahren des MAX-Knotens. Sobald der aktuelle Wert des MAX-Knotens grösser gleich β wird, kann die Auswertung abgebrochen werden. Deshalb ist β eine obere Grenze.

Diese Abschneide-Grenzen α und β werden bei jeder Änderung der Werte nachgeführt. Wenn also ein Vaterknoten einen neuen Wert bekommt, wird dieser allen seinen Nachfolger übermittelt.

Beispiel 1: α-bound

Bild 3: Die Evaluation des Knotens H. H ist ein MIN-Knoten. Der grösste Wert aller seiner MAX-Vorfahren ist der Wert der Wurzel (ROOT). Dieser Wert ist 10. Die α-Grenze ist nach der α-bound-Regel also 10.
Sobald nun der Knoten H den Wert 5 von seinem linken Nachfolger bekommt, ist die α-Grenze unterschritten. H gibt dies seinem Vorgänger I bekannt und dieser fährt mit der Evaluation seines anderen Nachfolgers K fort. I übergibt K ebenfalls die α-Grenze 10.
Hätte aber H zum Beispiel den Wert 15 anstatt 5 an I geliefert, so hätte I neu diesen Wert bekommen, da 15 grösser als 10 ist. Gleichzeitig wäre 15 auch die neue α-Grenze für K geworden!

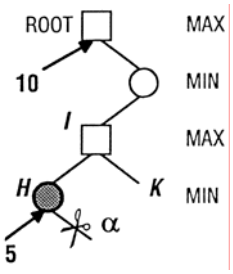


Bild 3 – α-bound

Beispiel 2: β-bound

Bild 4: Die Evaluation des Knotens G. G ist ein MAX-Knoten. Der kleinste Wert aller seiner MIN-Vorfahren ist 10 (Knoten C) und dieser wird damit auch zur β-Grenze. Die Auswertung des Knotens G ergibt nun als erstes den Wert 14 von seinem linken Nachfolger. Aber schon die 14 übersteigt die β-Grenze 10. Eine weitere Evaluation von G ist deshalb nicht mehr nötig. G meldet dies seinem Vorgänger C. Da C keine weiteren Nachfolger mehr hat, ist sein Wert nun definitiv 10 und er gibt diesen Wert weiter nach oben.

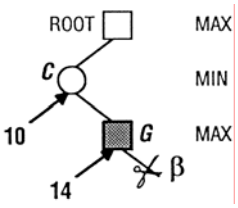


Bild 4 – β-bound

Wenn wir die beiden Regeln nun auf den ganzen Spielbaum anwenden, dann sieht er schlussendlich folgendermassen aus (Bild 5):

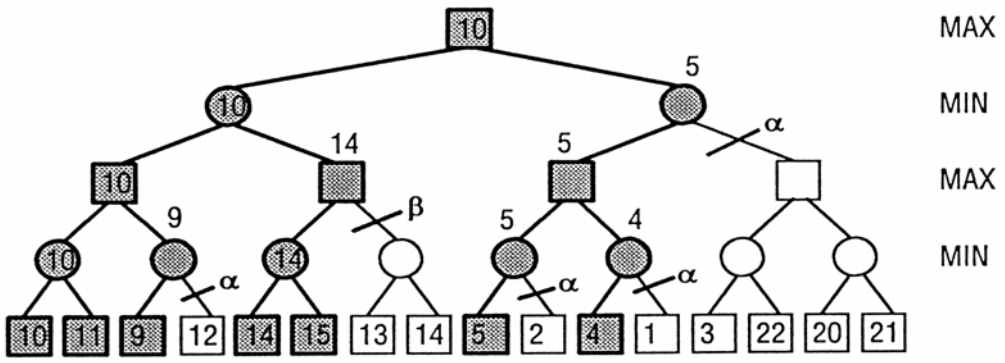


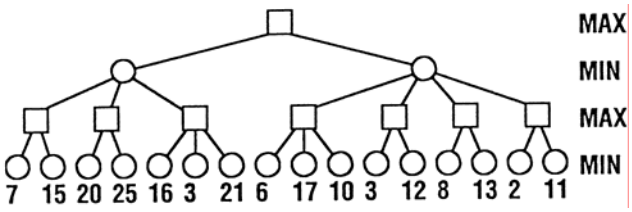
Bild 5 – Beispiel einer Spielbaum-Auswertung mit α-β pruning

Schüler-Lernkontrolle

Serie A

Diese Kontrollaufgaben sollen Ihnen zeigen, ob Sie den Stoff beherrschen. Sie sollen ja zu einem Experten auf Ihrem Gebiet werden, damit Sie Ihre Kollegen nachher auch kompetent unterrichten können!
Bearbeiten Sie die folgenden Aufgaben schriftlich. Sie sollen die Aufgaben alleine lösen. Sie dürfen die Unterlagen dazu nicht benutzen. Wenn Sie fertig sind, können Sie bei mir die Lösungen anschauen und mit den eigenen vergleichen. Haben Sie falsche oder unvollständige Lösungen aufgeschrieben, müssen Sie die entsprechenden Stellen im Text nochmals genau studieren.

Aufgabe 1



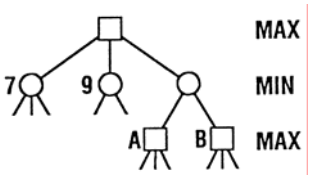
Nebenstehend ist ein Spielbaum mit Suchtiefe drei abgebildet. Berechnen Sie den Wert der Wurzel mit Hilfe der α-β-Methode. Streichen Sie dabei alle Knoten durch, die dank α-β nicht berechnet werden müssen. Schreiben Sie jeweils daneben ein α oder β, je nachdem, welche der beiden Regeln Sie angewandt haben.
Tip: α-Regel wird bei MIN-Knoten angewandt, die β-Regel bei MAX-Knoten.

Aufgabe 2

Formulieren Sie mit eigenen Worten die α-Regel.

Aufgabe 3

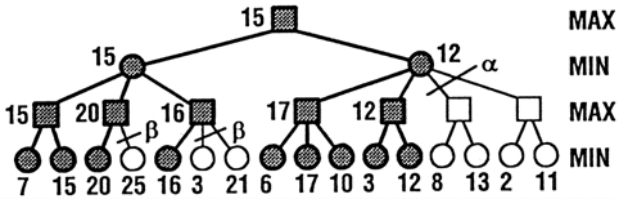
Geben Sie den Bereich an, in welchem der Wert von A liegen muss, damit B und seine Nachfolger nicht mehr berechnet werden müssen.



Lösungen

Serie A

Aufgabe 1



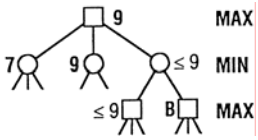
So sollte Ihre Lösung etwa aussehen. Wichtig ist, dass Sie die richtigen Äste abgeschnitten haben. Wenn Sie irgendwo zuviel oder zuwenig abgeschnitten haben, dann versuchen Sie mit Hilfe von Ihren Unterlagen, die richtige Lösung zu verstehen. Falls Sie nicht durchkommen, dann fragen Sie einen Kollegen aus Ihrer Expertengruppe.

Aufgabe 2

Vergleichen Sie Ihre Lösung mit der Regel in Ihren Unterlagen. Folgende Punkte sollten in Ihrer Regel vorhanden sein:

- Die α-Regel wird auf MIN-Knoten angewandt.
- Die α-Grenze ist eine untere Grenze, d.h. sobald sie unterschritten wird, kann die Suche abgebrochen werden.
- α ist das Maximum aller MAX-Vorfahren.

Aufgabe 3



Wenn A einen Wert hat, der kleiner oder gleich neun ist, wird sein MIN-Vorfahre somit höchstens den Wert neun erhalten. Damit ergibt sich kein grösserer Wert für den Wurzelknoten und B muss deshalb nicht mehr evaluiert werden.

Vier interessante Begriffe in der Spieltheorie

Übersicht

In diesem Teil des Puzzles lernen Sie einige neue Begriffe aus der Welt der Spieltheorie kennen. Sie werden dabei an Probleme aber auch an neue Strategien zur Verbesserung des Computerspieles herangeführt.

Vorgehen

1. Studieren Sie den Lehrtext "Vier interessante Begriffe ...". (Zeitaufwand: ca. 45 Min.)
2. Lösen Sie nun die Aufgaben der "Schüler-Lernkontrolle". (Zeitaufwand: ca. 15 Min.)
3. Besprechen Sie mit den anderen Experten Ihres Themas, wie ihr eure Mitschüler unterrichten wollt. (Zeitaufwand: ca. 30 Min.)
4. Falls Sie zusätzliche Vorbereitungen für die Unterrichtsrunde brauchen, erledigen Sie diese als Hausaufgabe.

Lernziele für die ganze Klasse

1. Sie können einem Laien die Begriffe Horizont-Effekt, Search-Extension und kombinatorische Explosion erklären.
2. Sie wissen, wie die Tiefensuche funktioniert und können erklären, was ihr grosser Vorteil ist.

Vier interessante Begriffe in der Spieltheorie:

Horizont-Effekt, Search-Extension, kombinatorische Explosion, Breiten- versus Tiefensuche

1. Repetition: Spielbaum

In der Einführung in die Spieltheorie haben Sie davon gehört, dass man alle möglichen Spielzüge mit Hilfe eines Spielbaumes darstellen kann (Bild 1). Die Wurzel stellt den aktuellen Spielstand, z.B. eines VIER GEWINNT, dar. Falls nun Schwarz am Zug ist, hat er sieben verschiedene Züge zur Auswahl. Nach jedem Zug ist das Spiel in einem anderen Zustand. Dieser Zustand wird durch einen Knoten dargestellt. Von diesem Zustand aus hat nun der weisse Spieler seinerseits auch sieben Zugsmöglichkeiten. So entsteht nun ein (auf den Kopf gestellter) Baum mit allen möglichen Zügen. Einfachheitshalber wird dieser Baum oft nicht ganz aufgezeichnet, sondern schematisch als Dreieck dargestellt (Bild 2).

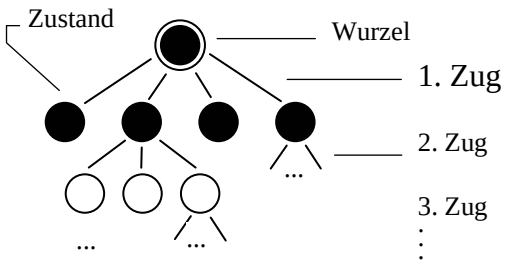


Bild 1 - Spielbaum

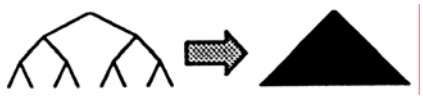


Bild 2 – Darstellung eines Spielbaumes

1. Horizont-Effekt

In der Einführung haben Sie einige neue Begriffe kennengelernt. Einer davon war der Ausdruck Suchtiefe. Die Suchtiefe gibt darüber Auskunft, wie viele Spielzüge der Spielcomputer zum Voraus berechnet. Wenn also zum Beispiel die Suchtiefe vier beträgt, dann bedeutet das,

dass der Rechner vier Züge vorausblickt, um seinen besten Zug zu finden. (So nebenbei: Die schnellsten Schachcomputer (1993) können bei einer normalen Spielstellung im Mittelspiel ca. 10 Züge vorausdenken, die weltbesten Spieler nur etwa 3-5 Züge!)
Im Beispiel "sieht" der Computer vier Züge weit. Wir Menschen nennen den Ort, bis wohin wir sehen können, den Horizont. Es wird deshalb häufig der Ausdruck Horizont verwendet, wenn es um die vorgegebene Suchtiefe geht.
Durch den Horizont (oder die Suchtiefe) ergibt sich aber ein Nachteil, den ich mit dem folgenden Beispiel beim VIER GEWINNT zeigen möchte (Bild 3):

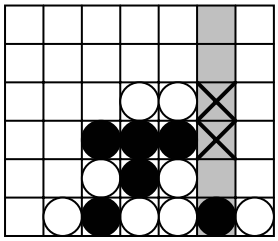


Bild 3 – Spielstand 1

Der Spieler mit den schwarzen Steinen hat sich eine hundertprozentige Gewinnchance herausgespielt. In der zweiten Kolonne von rechts hat er zwei Felder (mit X bezeichnet), die zu seinem Sieg führen, wenn er es schafft, dort einen schwarzen Stein zu legen. Da diese beiden Felder aber direkt übereinander liegen, ist der Sieg für Schwarz gewiss: Wenn er seinen Stein in das untere Feld legen kann, hat er gewonnen. Falls ihm aber Weiss zuvorkommt, gewinnt er mit dem nächsten Stein im oberen Feld.
Damit es aber soweit kommt, muss zuerst ein Spieler (Schwarz oder Weiss) seinen Stein in diese Kolonne spielen. Der weisse Spieler wird dies sicher nicht tun, weil er sonst beim nächsten Zug seines Gegners verliert. Das bedeutet also, dass der schwarze Spieler diesen Zug machen muss. Nun kommen wir aber zum Haken, den der Horizont haben kann!
Nehmen wir an, dass der Computer mit den schwarzen Steinen spielt. Wir stellen zudem seine Suchtiefe auf zwei ein. Er schaut also zwei Züge voraus: seinen eigenen (schwarzen) und den nächsten seines Gegners (weiss). Um zu gewinnen, müsste Schwarz ja seinen Stein in der sechste Kolonne von links legen (Bild 4, Spielstand 2). Danach wird Weiss natürlich auch dort spielen, um nicht direkt zu verlieren (Spielstand 3). Da Schwarz nun an seinem Such-Horizont angelangt ist, wertet er diese Stellung aus.

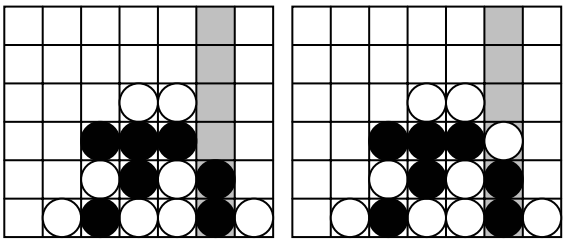
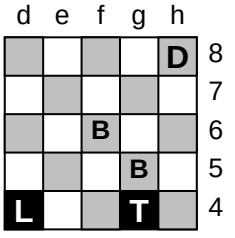


Bild 4 – Spielstände 2 und 3

Seine Stellung hat sich verschlechtert, denn die horizontale schwarze 3er-Reihe hat nur noch ein anstatt zwei offene Enden. Zudem hat der weisse Spieler jetzt neu drei Steine in einer Reihe. Diese Stellung scheint also für Schwarz ungünstig und er wird deshalb seinen Stein nicht in die sechste Kolonne legen. Er vergibt somit einen sicheren Sieg, weil er zu wenig weit sieht.
Gut zu spielen kann deshalb auch bedeuten, dass man zuerst einen Zug spielen muss, der auf den ersten Blick schlecht aussieht, damit man nachher umso mehr gewinnen kann. Normalerweise sieht der Computerspieler genug weit voraus, um zu erkennen, ob ein im Moment schlechter Zug zu einer besseren Stellung führen kann. Problematisch wird es erst, wenn der gute Zug jenseits des Horizonts liegt, wenn ihn also der Computer nicht mehr sieht. Man spricht in diesem Fall vom Horizont-Effekt.
Im VIER GEWINNT tritt der Horizont-Effekt selten auf und der obige Fall ist schon eher ein gesuchtes Beispiel. Im SCHACH jedoch kommt er häufig vor, meistens dann, wenn es zu einem Schlagabtausch kommt. Bei diesem verliert zuerst der eine Spieler eine Figur. Danach kann er aber eine gegnerische Figur schlagen. Ein solcher Schlagabtausch kann einen grossen Gewinn für mich bedeuten, wenn ich z.B. meinen Turm opfere, aber dafür die gegnerische Dame



1. S: Tg4 x g5
   W: Bfg x g5
   Bewertungsfunktion ergibt Turmverlust

Horizont

2. S: L x Dh8
   Damengewinn.

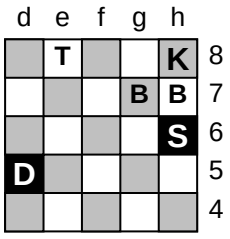
Bild 5 – Turmverlust vs. Damengewinn

schlagen kann (Bild 5). Aber auch hier kann mich mein Horizont daran hindern, den Turm zu opfern, wenn ich nicht sehe, dass ich nachher die gegnerische Dame schlagen kann. Dieses Problem wurde schon früh entdeckt. Man hat deshalb nach einer Lösung Ausschau gehalten. Wie heute der Horizont-Effekt umgangen wird, erfahren Sie im nächsten Abschnitt.

2. Search-Extension

Aus dieser englischen Wortkomposition lässt sich schon herauslesen, um was es in diesem Abschnitt geht: Search-Extension = Erweiterung der Suche. Der Computer soll also nicht nur bis zu der vorgegebenen Grenze (Suchtiefe/Horizont) den besten Zug finden, sondern noch weitersuchen. Es stellt sich nun die Frage: Wo soll der Computer weitersuchen und wo nicht?

Überall weiterzusuchen ist sicher nicht die Idee der Search-Extension. Vielmehr soll dort weitergesucht werden, wo sich eine Spielwende, schnelle Veränderungen oder gar das Spielende abzeichnet. Man nennt solche Zustände an der Suchgrenze als unruhige Stellungen.



2. S: Dd5 x g8 +
W: Te8 x Dg8

Horizont

dank Search-Extension bei unruhiger Stellung:

2. S: Sk6 x f7 ++
Schachmatt.

Bild 6 –Schach-Extension bei unruhiger Stellung

Einige schöne Beispiele von solchen unruhigen Stellungen findet man im SCHACH. Dort ist der Verlust einer Figur ist sicher eine solche Situation, weil darauf ein Schlagabtausch folgen könnte. Aber auch wenn Schach ist, sollte die Situation genauer untersucht werden. Deshalb lohnt sich auch in diesem Fall eine Weitersuche (Bild 6).

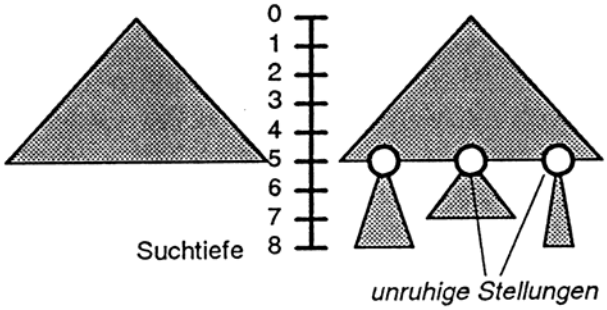


Bild 7

a) normaler Spielbaum b) Search Extension

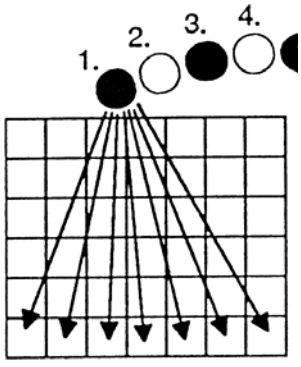
Es gilt also, unruhige Stellungen an der Suchgrenze zu erkennen und dort die Suchtiefe zu erhöhen, denn vielmals ist in solchen Stellungen einiges zu gewinnen ... oder zu verlieren!

In Bild 7 ist links ein Spielbaum zu sehen, der auf der ganzen Breite bis zur Tiefe fünf durchsucht wurde. Rechts ist der gleiche Baum dargestellt, bei dem aber an drei unruhigen Stellungen noch zwei bis drei Züge weitergesucht wurde.

3. Kombinatorische Explosion

Sie haben schon in der Einleitung davon gehört, dass die Breite eines Spielbaumes sehr schnell zunimmt, wenn man seine Tiefe vergrössert. Dies ist am Beispiel des VIER GEWINNT sehr schön zu sehen:

Zug-Nr	Anz. Mögl.	Totale Anz. Mögl.		
1	7	7	=	7
2	7	7*7	=	49
3	7	7*7*7	=	343
4	7	7*7*7*7	=	2'401
5	7	7*7*7*7*7	=	16'807
6	7	7*7*7*7*7*7	=	117'649



7 7 7*7*7*7*7*7*7 = 823'543

Die mittlere Spalte des Spieles ist vielleicht nach fünfzehn Zügen voll, d.h. bis dahin gibt es für jeden Zug sieben Möglichkeiten, einen Stein zu legen. Die totale Anzahl Mögliche Spielzüge bis zum 15. Zug beträgt also 4'747'561'509'940 ≈ $4.7 \cdot 10^{12}$! Da die Anzahl der möglichen Spielstellungen (Kombinationen) mit der Suchtiefe so rasant zunimmt, spricht man von einer kombinatorischen Explosion.
Im VIER GEWINNT habe ich, wenn ich drankomme, maximal "nur" sieben mögliche Züge zur Auswahl. Beim SCHACH gibt es aber schon am Anfang zwanzig verschiedene Züge, denn jeder Bauer und jeder Springer hat je zwei Zugsmöglichkeiten. Im Verlauf des Spieles steigt dann die Anzahl der möglichen Spielzüge sogar noch. Wenn man dieselbe Rechnung wie beim VIER GEWINNT nun beim SCHACH mit zwanzig Möglichkeiten macht, erhält man folgende Zahlen: Nach fünf Zügen gibt es schon über 1 Million und mit sieben Zügen schon über einer Milliarde Möglichkeiten! Da kommt auch ein schneller Rechner bald an seine Grenzen!

Bsp: Wir nehmen an, dass die Feldbewertung eines Spielzustandes eine Millisekunde (1/1000s) benötigt. Dann braucht der Computer zur Berechnung eines Zuges !

4. Breiten- versus Tiefensuche

4.1. Breitensuche

In diesem Abschnitt geht es um die Suche des besten Zuges vom aktuellen Spielstand aus. Diesen Zug wollen wir mit Hilfe eines Spielbaumes herausfinden. Dazu muss der Baum mit einer geeigneten Methode durchsucht werden. Die erste Möglichkeit ist die Breitensuche (Bild 8):

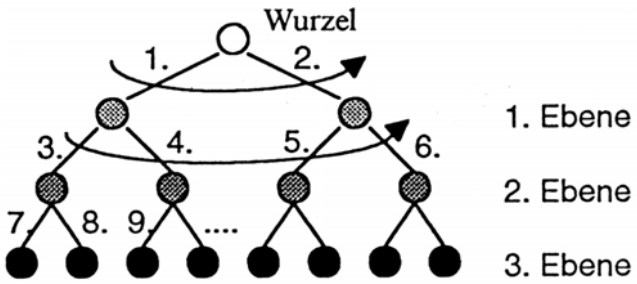


Bild 8 – Breitensuche

Vom aktuellen Spielstand (Wurzel) aus suche ich alle möglichen Spielzüge. So erhalte ich alle Nachfolgezustände der Wurzel. Diese Zustände liegen alle auf der ersten Ebene. Als nächstes generiere ich alle Züge der zweiten Ebene, indem ich von jedem Zustand der ersten Ebene alle seine Nachfolgezustände suche. Dies mache ich solange, bis ich bei der vorgegebenen Suchtiefe angelangt bin. Diese Art von Suche heisst Breitensuche, weil ich von Anfang an auf der ganzen Breite des Baumes nach möglichen Zügen suche.

4.2. Tiefensuche

Anders verhält es sich bei der Tiefensuche. Hier suche ich mir den erstbesten Zug von der Wurzel aus und gelange so zum nächsten Zustand (Bild 9). Auch von jedem weiteren Knoten suche ich nur einen Nachfolgezug und gehe von da zum nächsten Zustand weiter in die Tiefe. Erst wenn ich auf der Suchtiefe angekommen bin, suche ich nach Zügen auf gleicher Ebene.

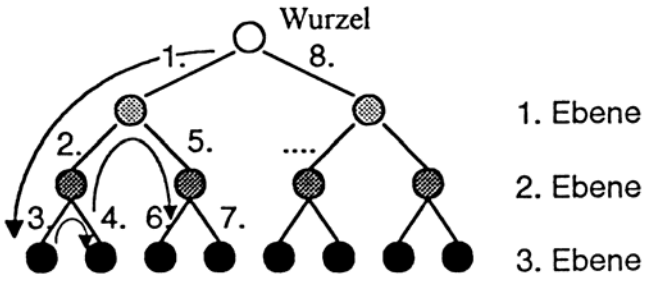


Bild 9 – Tiefensuche

4.3. Welche Suche soll nun in der Praxis verwendet werden?

Jeder Spielstand, von dem ich noch Nachfolgezüge suchen muss, braucht irgendwo gespeichert zu werden. Erst wenn alle diese Züge generiert und ausgewertet wurden, kann der Spielzustand wieder gelöscht werden. Anhand des folgenden kleinen Beispiels werden Sie sehen, welche der beiden Suchen brauchbar ist und welche nicht benutzt werden sollte:

Beispiel: Wir spielen VIER GEWINNT gegen den Computer. Er ist am Zug und soll den besten Zug herausfinden. Wir lassen ihn fünf Züge vorausrechnen und nehmen an, dass es bei jedem Zug sieben Möglichkeiten gibt.

Breitensuche: Die unterste Ebene wird erst am Schluss ausgewertet. Das bedeutet, dass alle Spielzustände bis und mit der zweituntersten Ebene bis zu diesem Zeitpunkt gespeichert werden müssen. Die zweitunterste Ebene ist die vierte Ebene.

1. Ebene:	7 Zustände	=	7	
2. Ebene:	7*7 Zustände	=	49	
3. Ebene:	7*7*7 Zustände	=	343	
4. Ebene:	7*7*7*7 Zustände	=	2401	
Total müssen also gespeichert werden:			2800	Zustände

Tiefensuche: Beim Tiefergehen im Baum muss pro Ebene nur ein Spielzustand gespeichert werden, denn die Werte der Knoten auf einer Ebene werden der Reihe nach berechnet. Wenn also ein Knoten evaluiert ist, kann der Platz im Speicher wieder freigegeben werden, da der berechnete Wert an den Vaterknoten übermittelt wird.
Die Zustände auf der untersten Ebene müssen auch bei dieser Suche nicht gespeichert werden. Das bedeutet also, dass zu jedem beliebigen Zeitpunkt höchstens Speicherplatz für 4 Zustände benötigt wird (Bild 10)!

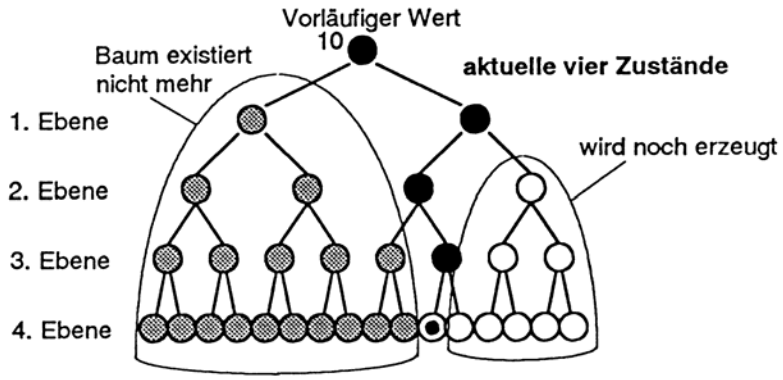


Bild 10 – Tiefensuche

Fazit: Breitensuche ist unbrauchbar, weil es zu viel Speicherplatz benötigt!

Schüler-Lernkontrolle

Serie A

Diese Kontrollaufgaben sollen Ihnen zeigen, ob Sie den Stoff beherrschen. Sie sollen ja zu einem Experten auf Ihrem Gebiet werden, damit Sie Ihre Kollegen nachher auch kompetent unterrichten können!
Bearbeiten Sie die folgenden Aufgaben schriftlich. Sie sollen die Aufgaben alleine lösen. Sie dürfen die Unterlagen dazu nicht benutzen. Wenn Sie fertig sind, können Sie bei mir die Lösungen anschauen und mit den eigenen vergleichen. Haben Sie falsche oder unvollständige Lösungen aufgeschrieben, müssen Sie die entsprechenden Stellen im Text nochmals genau studieren.

Aufgabe 1 - Breitensuche
Jemand schreibt ein Spielprogramm, das den besten Zug mit Hilfe von Breitensuche ermittelt. Der Computer, auf dem das Programm installiert ist, kann 20'000 Spielzustände speichern. Wieviele Spielzüge kann der Computer vorausberechnen, wenn es pro Zug durchschnittlich 11 Zugsmöglichkeiten gibt?

Aufgabe 2 - Search-Extension
Erklären Sie in 2-3 Sätzen, warum sich die Search-Extension (Erweiterung der Suche) lohnt.

Aufgabe 3 - Horizont-Effekt / Search-Extension
Wir nehmen an, Sie hätten ein VIER GEWINNT programmiert, das sechs Züge vorausrechnet (das ergibt 7^6 Berechnungen). Sie wollen nun neu in Ihrem Programm den Horizont-Effekt berücksichtigen und bauen deshalb eine Search-Extension ein. Da Sie für die Search-Extension wieder mehr Züge berechnen müssen, der Computer aber gleich schnell spielen soll, setzen Sie die Suchtiefe zurück auf fünf.
Um wieviele Züge können Sie die Suche erweitern (Search Extension), wenn wir davon ausgehen, dass es am Horizont fünf unruhige Stellungen hat? (Annahme: Es kann jeweils in alle sieben Spalten gespielt werden.)

Lösungen

Serie A

Aufgabe 1 - Breitensuche

1. Zug	11	=	11	1. Zug	11
2. Zug	11^2	=	121	1.+ 2. Zug	132
3. Zug	11^3	=	1'331	1. - 3. Zug	1'463
4. Zug	11^4	=	14'641	1. - 4. Zug	16'104
5. Zug	11^5	=	161'051	1. - 5. Zug	177'155

Der Computer kann die Züge bis zum vierten Zug speichern (16'104 < 20'000). Das heisst also, dass er alle fünften Züge noch auswerten kann, weil er dazu die Spielstellungen nicht mehr speichern muss. Die Suchtiefe beträgt also 5!

Aufgabe 2 - Search-Extension

Ihre Lösung sollte etwa die folgenden Punkte enthalten:

- Unruhige Stellungen können besser bewertet werden, wenn noch einige weitere Züge beachtet werden.
- Der Horizont-Effekt kann umgangen werden.
- Der Computerspieler spielt in kritischen Situationen besser.

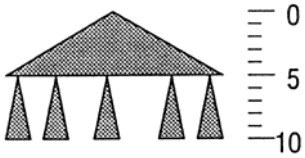
Aufgabe 3 - Horizont-Effekt / Search-Extension

Lösung: Search-Extension um 5 Züge.

Anzahl Berechnungen, die für Search-Extension eingesetzt werden können:

$$7^6 - 7^5 \approx 100'000$$

An fünf Stellen soll die Suche erweitert werden: Es stehen also 100'000 : 5 = 20'000 Berechnungen pro Search-Extension zur Verfügung.



Die Anzahl Berechnungen in einem Baum betragen für die Suchtiefen 5 und 6:

$$7^5 = 16'807 \quad 7^6 = 117'649$$

16'807 < 20'000, d.h. die Suchtiefe in den Bäumen beträgt 5.

Heuristik

Übersicht

In diesem Kapitel wollen wir uns mit Funktionen beschäftigen, die wir mathematisch nicht darstellen können. In der Spieltheorie werden oft Abbildungen gebraucht, die zwar beweisbar existieren und berechenbar sind, aber von einem Computer nicht in nützlicher Frist berechnet werden können. Ein Beispiel dazu ist die Feldbewertungsfunktion: Jeder Spielstellung soll ein Wert zugeordnet werden, der aussagt, welche Gewinnchancen der Spieler mit den weissen Figuren hat. Diese Funktion ist eindeutig, denn wenn beide Spieler ideal spielen wird entweder Weiss gewinnen, oder Schwarz wird gewinnen, oder es wird ein Remis geben. Doch diese Funktion ist nur unter einem riesigen Aufwand zu berechnen (mehrere Jahrtausende für Schach).

Vorgehen

Wissenserwerb:
Studiere die vorliegenden Unterlagen sorgfältig. Wenn Du das Gefühl hast, Du hast den Stoff begriffen, so löse die Aufgaben der Schüler-Lernkontrolle ganz am Ende.
Expertenrunde:
Besprecht in der Gruppe, was Ihr Euren MitschülerInnen mitteilen wollt. Beachtet dazu die unten angegebenen Lernziele!Überlegt Euch, wie Ihr diesen Stoff den ZuhörerInnen vermittelt wollt.

Lernziele

Nach dem Durcharbeiten dieses Kapitels weisst Du, in welchen Fällen in der Informatik heuristische Methoden eingesetzt werden. Du verstehst, warum Heuristik überhaupt eingesetzt wird, da doch andere Methoden bessere Resultate liefern.
Konkret begreifst Du, wie eine Feldbewertungsfunktion für ein beliebiges Spiel gesucht werden kann. Du bist auch in der Lage, ein Minimax-Programm so zu erweitern, dass es nur noch einen Teil der möglichen Züge testet, also den Spielbaum nur partiell absucht.

Material

- "Bauernschach", eine Einführung in ein vereinfachtes Schachspiel

Heuristik

Im Informatikduden steht unter dem Stichwort "Heuristik" die folgende Beschreibung:

> "**Heuristik**: Algorithmen zur Lösung komplexer Probleme verbessert man häufig durch Strategien, die oft auf Hypothesen und Vermutungen aufbauen und die mit höherer Wahrscheinlichkeit (jedoch ohne Garantie) das Auffinden einer Lösung beschleunigen sollen. Solche Strategien heißen *Heuristiken*. Faustregeln, bereits früher beobachtete Eigenschaften von Lösungen oder die Nachbildung des menschlichen Problemlösungsprozesses sind typische Heuristiken."

Eine heuristische Funktion ist also eine Funktion, die nicht auf wissenschaftlichen Erkenntnissen beruht. Wir sind voll und ganz auf unsere Intuition gestellt. Ausserdem können wir eine heuristische Funktion nicht werten. Es gibt nicht ein richtig und ein falsch. Wir können eine Funktion höchstens an Beispielen messen und dann Aussagen über ihr Verhalten bei diesen Beispielen machen. Die eine Funktion wird sich vielleicht eher unseren Hoffnungen entsprechend verhalten als die andere.
In der Spieltheorie kann Heuristik an zwei verschiedenen Orten verwendet werden. Zum einen kann es sinnvoll sein, bei einer bestimmten Spielsituation nicht alle möglichen Gegenzüge durchzutesten, also eine heuristische Auswahl aus den möglichen Zügen zu treffen. Zum andern kann im allgemeinen ein Spiel nicht bis zum Ende probeanalysiert werden, da dies zu einem enormen Aufwand führen würde. Also muss der Computer in der Lage sein, eine Spielstellung zu bewerten. Diese Bewertungsstrategie ist ebenfalls heuristischer Natur, da niemand Kriterien kennt, wie eine solche Spielstellung[1] bewertet wird. Ausnahme bilden einige sehr einfache Spiele wie zum Beispiel **Nimm**, bei denen mit Hilfe einer Formel entschieden werden kann, ob eine Stellung zum Sieg oder zur Niederlage führt.Um über Suchstrategien im Spielbaum sprechen zu können, müssen wir vorab einige Abmachungen treffen. Insbesondere wollen wir uns über die folgenden Definitionen einigen:

d = deepth = Suchtiefe = Anzahl (Halb[2]-) Züge vordenken.
b = breadth = Suchbreite = durchschnittliche Anzahl möglicher Züge.

Die Suchbreite b ist durch das Spiel gegeben! Beim sieben Spalten breiten **Vier Gewinnt** mit runterfallenden Steinen ist b≈7. Zu Beginn beträgt die Suchbreite exakt sieben, sie kann aber im Verlauf des Spieles abnehmen, sobald Spalten vollständig gefüllt sind.
Im Schach ist die Suchbreite b≈20. Das bedeutet, dass der ziehende Spieler seinen Zug unter etwa zwanzig möglichen Zügen auswählen kann.

Die Suchtiefe d hingegen ist frei wählbar. Je grösser d gewählt wird, desto stärker spielt der Computer. Mit einem unendlich grossen d spielt der Computer perfekt. Er analysiert den ganzen Spielbaum. Ein grosses d bewirkt aber auch eine lange Rechenzeit. In heutigen Programmen beträgt die Suchtiefe d so zwischen 3 und 12.

Die Anzahl der Züge, die der Computer analysieren (probespielen) muss, lässt sich mit der folgenden Formel berechnen:

$$\underbrace{b \cdot b \cdot \ldots \cdot b}_{d-mal} = b^d$$

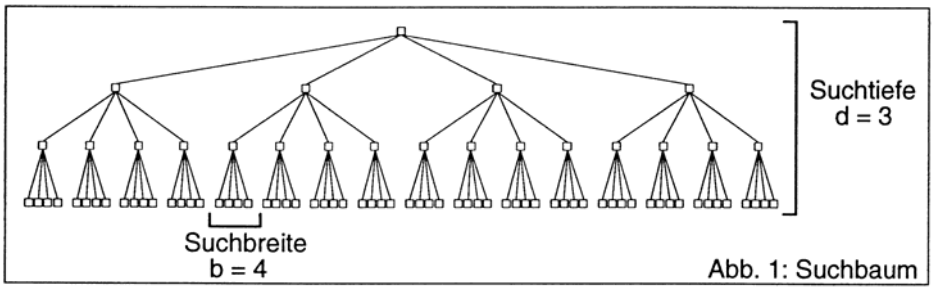


Abb. 1: Suchbaum

Siehe dazu auch den Spielbaum in der Abbildung 1. In diesem Beispiel wurde eine Suchtiefe von drei Zügen gewählt. Die Suchbreite beträgt vier. In den meisten realistischen Spielen ist die Suchtiefe viel grösser, so dass der Spielbaum viel schneller in die Breite wächst (man sagt auch, er "explodiert").

[1] Konstellation der Figuren und Spieler, der an der Reihe ist.
[2] Wir sprechen hier oft über „Züge“ und meinen eigentlich Halbzüge. Ein Halbzug ist der Zug eines Spielers, ein Ganzzug je ein Zug der beiden Spieler.

Auf der Stufe d muss also der Computer die Spielstellung bewerten. Dazu benötigt er eine Funktion

$$f : Spielstellung \rightarrow Zahl,$$

die einer guten Stellung eine grosse Zahl und einer schlechten Stellung eine kleine (oder eine negative) Zahl zuordnet. Eine solche Funktion heisst Feldbewertungsfunktion. Oft wird die Bewertung aus der Sicht des ziehenden Spielers berechnet. Hier in diesem Text wird die Spielstellung aber immer aus der Sicht von Weiss bewertet, um dem Wirrwarr Grenzen zu setzen.

Die Feldbewertungsfunktion

Eine Feldbewertungsfunktion hat die Aufgabe, eine Spielstellung zu bewerten. Je besser die Stellung ist, desto mehr Punkte soll sie vergeben. Korrekterweise müsste man von einer statischen Feldbewertungsfunktion sprechen. Statisch bedeutet, dass die Bewertungsfunktion nur die aktuelle Stellung betrachtet, aber nicht vordenkt. Die Funktion kennt weder die Vergangenheit noch die Zukunft des Spielablaufs.
Bevor wir entscheiden können, was eine gute Feldbewertungsfunktion ist, müssen wir festlegen, was eine gute Spielstellung ist. Die einfachste Möglichkeit ist die folgende:

- Eine gute Spielstellung ist eine Stellung, die zum Sieg führt
- Eine schlechte Spielstellung ist ein Stellung, die zur Niederlage führt.

Eine Feldbewertungsfunktion mit den obenstehenden Eigenschaften ist ideal. Ist hiermit die ganze Spieltheorie im Eimer?
Zwar ist beweisbar, dass eine solche Bewertungsfunktion existiert, aber sie ist im allgemeinen sehr komplex und kann kaum berechnet werden. Deshalb braucht man eine einfachere, probabilistische[3] Funktion. Diese Funktion ist ausserdem heuristisch, das heisst, man weiss nicht so recht, welche Funktionen gut sind und welche nicht. Wir müssen also eine Funktion "erfinden" und diese dann ausprobieren. Wenn sie gut ist, behalten oder verbessern wir sie, sonst "erfinden" wir eine andere.
Eine ganz einfache Feldbewertungsfunktion, die in fast allen Brettspielen Gültigkeit hat, ist die Materialbewertungsfunktion. Je mehr Figuren ein Spieler auf dem Spielbrett hat, desto mehr Punkte kriegt er. Diese Funktion ist aber sehr schlecht, da sie die Stellungsvorteile eines Spielers vollständig vernachlässigt! Also brauchen wir eine andere Funktion, die sowohl das Material als auch die Spielstellung eines Spieler beurteilt und eine Gesamtpunktzahl berechnet.

Wie weiter oben schon gesagt: Wir können nur heuristische Funktionen finden! Suchen wir nun gemeinsam eine Feldbewertungsfunktion für eine einfaches Spiel: **Bauernschach**.

Studiere dazu bitte das Dokument "Bauernschach".

Wir wollen nun anhand des Beispiels **Bauernschach** eine Feldbewertungsfunktion finden. Eine erste Idee ist ganz einfach:

[3] Funktion, die *wahrscheinlich* das richtige Resultat liefert.

$$f : Stellung \rightarrow \begin{cases} 1, \text{falls Weiss gewonnen hat} \\ -1, \text{falls Schwarz gewonnen hat} \\ 0, \text{falls Spiel noch nicht entschieden ist} \end{cases}$$

Diese Bewertungsfunktion taugt nicht viel! In den meisten Fällen wird sie 0 zurückgeben und damit nichts über einen Zug aussagen. Erst wenn bis zum Spielende vorgedacht werden kann, werden andere Werte als 0 zurückgegeben. Dann wird Kurs auf Sieg genommen.

Wir müssen also die Funktion verbessern. Ein möglicher Vorschlag ist der folgende:

$$f : Stellung \rightarrow \sum_{i=1}^{3} i \cdot (\text{Anzahl weisse Figuren in Zeile } i) - \sum_{i=1}^{3} (4-i) \cdot (\text{Anzahl schwarze Figuren in Zeile } i)$$

Dabei bekommt eine Figur umsomehr Wert, je weiter vorne sie auf dem Spielfeld steht. Der Wert der schwarzen Figuren wird abgezogen, da die Bewertung aus der Sicht von Weiss geschieht.
Diese Funktion ist schon ganz hübsch. Betrachten wir doch die folgende Programmierung in der Sprache **PASCAL**:

```
Wert := 0;
FOR i := 1 TO 3 DO                          (* 3 Zeilen *)
  FOR j := 1 TO 3 DO                        (* zu drei Spalten *)
    IF Feld[i,j] = "W" THEN                 (* weisse Figur *)
      Wert := Wert + i
    ELSE IF Feld[i,j] ="S" THEN             (* schwarze Figur *)
      Wert := Wert - (4-i);
(* Wert enthält nun die Stellungsbewertung aus der Sicht von Weiss *)
```

Die Funktion hat aber noch ein Manko: Sie erkennt nicht, wenn ein Spieler gewinnt. Eine zusätzliche Abfrage sollte im Falle eines Sieges von Weiss eine grosse Zahl (zum Beispiel +100) und im Falle eines Sieges von Schwarz eine kleine Zahl (zum Beispiel -100) zurückgeben.

Dieses Problem lässt sich aber auch einfacher lösen!

Zur Zeit gibt jede Figur in der ersten Zeile einen Punkt, in der zweiten Zeile zwei Punkte, und in der dritten Zeile drei Punkte. Damit die dritte Zeile viel mehr Punkte gibt als die ersten beiden Zeilen, können wir einfach die Punktzahlen quadrieren oder noch besser hoch drei nehmen!

$$1^3=1$$
$$2^3=8$$
$$3^3=27$$

Somit wird die Punktzahl beim Sieg (mindestens 27) grösser als jede ohne Sieg erreichbare Punktzahl (höchstens 24).

Die neue Feldbewertungsfunktion lautet:

$$f : Stellung \rightarrow \sum_{i=1}^{3} i^3 \cdot (\text{Anzahl weisse Figuren in Zeile } i) - \sum_{i=1}^{3} (4-i)^3 \cdot (\text{Anzahl schwarze Figuren in Zeile } i)$$

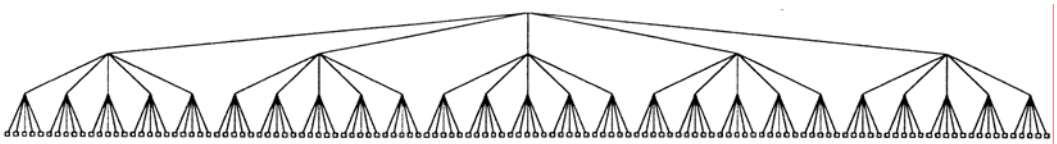
Das Programm ist ebenfalls entsprechend zu ändern.

Ist diese Funktion nun gut?
Bevor wir sie empirisch[4] testen, wollen wir uns überlegen, ob sie wirklich alle Fälle abdeckt. Si-cher-lich wird der Computer mit dieser Funktion versuchen, seine Bauern möglichst rasch nach vorne zu stossen. Auch wird er bemüht sein, keine Figur zu verlieren, denn eine fehlende Figur gibt keine Punkte. Aber wir haben abgemacht, dass der Gegner gewinnt, wenn ein Spieler nicht ziehen kann. Also sollte der Computer auch versuchen, den Gegner zu blockieren, beziehungsweise sich selber nicht blockieren zu lassen.

Dies wäre nun ein weiterer Schritt, wie wir die Funktion verbessern könnten. Dies wollen wir aber an dieser Stelle nicht mehr tun.

Partielle Suche

In der Minimax-Strategie muss der Computer auf jeder Stufe im Spielbaum alle möglichen Züge weiterverfolgen. Wenn wir zum Beispiel die Suchtiefe auf fünf Züge eingestellt haben, werden fünf Züge vorgedacht: Zu jedem weissen Zug wird jeder schwarze Zug getestet, zu jedem solchen schwarzen Zug wiederum jeder mögliche weisse Zug und so weiter. Das folgende Bild soll dieses Vorgehen verdeutlichen. Zuoberst ist die jetzige Spielstellung. Jede Linie steht für einen möglichen Zug, der weiterverfolgt werden muss. Jeder Knoten steht für eine Spielstellung. Ein Kästchen bedeutet, dass dieser Knoten mit der Feldbewertungsfunktion bewertet wird.

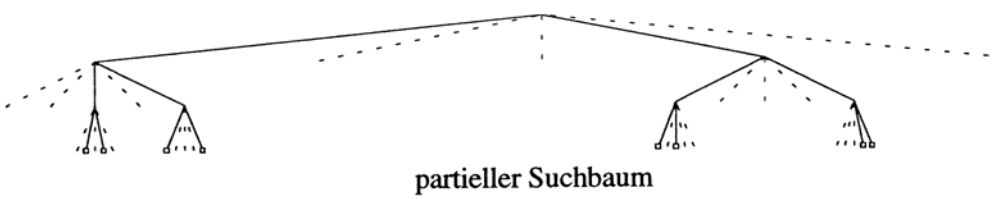


vollständiger Suchbaum

Nun könnte es doch sein, dass Weiss zehn Züge zur Auswahl hat, drei davon aber offensichtlich nichts taugen. Im **Schach** scheint es etwa unsinnig, eine Dame ungedeckt vor den gegnerischen König zu stellen.

Wenn nicht alle Züge weiterverfolgt werden, sprechen wir von einer *partiellen Suche*. Die Art und Weise, wie die Züge ausgesucht werden, denen weitere Beachtung geschenkt wird, spielt dabei keine Rolle. Auf jeden Fall ist diese Auswahl heuristisch. Bis heute ist kein Algorithmus und keine Strategie bekannt, wie man unsinnige Züge erkennen kann.

Betrachten wir den partiellen Suchbaum im folgenden Bild. In jeder Ebene werden nur zwei Züge weiterverfolgt. Diejenigen Züge, die nicht weiterverfolgt werden, sind im Bild durch gepunktete Linien dargestellt. Wie vorhin bedeutet ein Kästchen, dass dieser Knoten bewertet wird.



partieller Suchbaum

Dieses Bild zeigt, dass die Anzahl Feldbewertungen (Kästchen) drastisch reduziert werden kann. Die Frage, welche Züge denn nun weiterverfolgt werden sollen und welche nicht, ist aber immer noch unbeantwortet.

[4] an Beispielen ausprobieren

Wir wollen einige Vorschläge bearbeiten, wie diese Auswahl getroffen werden könnte. Wie schon weiter oben gesagt, Regeln oder Formeln existieren nicht. Wir müssen also mit Heuristiken arbeiten.

- Die besten fünf (oder n) Züge werden weiterverfolgt.
- Die besseren 50 Prozent (oder p %) der Züge werden ausgewählt.
- Es werden alle Züge beachtet, die höchstens k Punkte schlechter gewertet wurden als der beste Zug.

Bei all diesen Verfahren kann noch ein zusätzliches Element eingebaut werden: der Zufall. Zum Beispiel könnte nach einer Selektion noch ein Zug zufällig aus den ausgemusterten Zügen ausgewählt werden, der auch weiterverfolgt wird.

Nun stellt sich eine neue Frage: Wie können wir die möglichen Züge nach ihrer Güte ordnen. Wie können wir entscheiden, ob ein Zug besser ist als ein anderer?
Wir müssen diese Frage beantworten, wenn wir nur die besten fünf Züge weiterverfolgen wollen. Oder die besten 50 Prozent. Also brauchen wir ein schnelles Verfahren, das jedem Zug eine Punktzahl gibt. Dies funktioniert wie folgt:
Jeder Zug wird probeweise gezogen. Die entstandene Stellung wir bewertet und die Punktzahl dem Zug zugeordnet. Diejenigen Züge mit den besten Punktzahlen werden weiterverfolgt.
Zeichnen wir also das Bild von oben neu. Es kommen nämlich neue Kästchen dazu. Diese die-nen nicht dem Minimax-Algorithmus, sondern nur dem Auswahlverfahren!

partieller Suchbaum

Nun wollen wir den Aufwand des vollständigen Minimax-Algorithmus mit dem des partiellen vergleichen. Dazu berechnen wir die Anzahl Feldbewertungen, die nötig sind, um einen Zug zu spielen. Um die Rechnung durchführen zu können, gehen wir von idealen Bedingungen aus. Die Suchbreite sei konstant b, die Suchtiefe d, und im partiellen Algorithmus werden jeweils n Züge weiterverfolgt (wobei $n \leq b$).

Suchart	vollständig	partiell
Anzahl Knoten auf der Ebene i	b^i	$n^{i-1} \cdot b$
Wieviele Knoten der Ebene i (i < d) werden bewertet?	keine	alle
Wieviele Knoten der Ebene d werden bewertet?	alle	alle
Totaler Aufwand (Anzahl Bewertungen)	b^d	$\sum_{i=1}^{d} n^{i-1} \cdot b$ $= \frac{n^d - 1}{n-1} \cdot b$

Das Aufwandverhältnis beträgt also

$$v = \frac{\text{partieller Aufwand}}{\text{vollständiger Aufwand}} = \frac{\frac{n^d - 1}{n-1} \cdot b}{b^d}$$

Hier einige Beispiele:
Vier Gewinnt: b=7, d=5, n=4 $\Rightarrow$ 14%
Schach: b=20, d=8, n=18 $\Rightarrow$ 51%
Aber bei n=b:

Vier Gewinnt: b=7, d=5, n=7 $\Rightarrow$ 117%

Die Folgerung: Wenn wir einen partiellen Suchalgorithmus verwenden, müssen wir dafür sorgen, dass wirklich n<b ist! Sonst wird dieser Algorithmus langsamer als ein normales (vollständiges) Minimax! Auswahlheuristiken, die nur ab und zu einen Zug "rausschmeissen", im allgemeinen aber alle Züge weiterverfolgen, sind kontraproduktiv!

Und noch etwas:
Partielle Suche ist eine echte Einschränkung gegenüber der vollständigen Suche. Ein Programm mit einer vollständigen Minimax-Strategie spielt besser als ein partielles! Wir können zwar Zeit gewinnen, verlieren aber Spielstärke. Oft geht gerade die Angriffslustigkeit eines Programmes verloren. Partielle Suchverfahren scheinen etwas gegen Opfer zu haben. (warum?)

Das Vorgehen
Wir wollen hier kurz aufzeigen, wie denn so eine partielle Suche aussehen könnte. Zuerst betrachten wir ein vollständiges Minimax-Programm.

```
PROCEDURE Minimax (Tiefe,Spieler: INTEGER;
                   VAR Wert: INTEGER; VAR Zug: ...);
(* Tiefe = Suchtiefe, Spieler = 0/1, Wert = Feldbewertung *)
VAR TestZug,BesterZug: ...; BesterWert: INTEGER;
BEGIN
  WHILE "Es gibt noch mögliche Züge" DO
  BEGIN
    TestZug := "möglicher Zug";
    "Ziehe den Zug TestZug";
    IF Tiefe > 1 THEN
      Minimax (Tiefe-1, 1-Spieler, Wert, Zug);    (* Rekursiver Aufruf *)
      (* Tiefe nur noch eins weniger, anderer Spieler ist an der Reihe *)
    ELSE Wert := "Bewerte das aktuelle Feld";
    "Nimm den Zug TestZug zurück";
    IF Wert > BesterWert THEN
    BEGIN BesterWert := Wert; BesterZug := TestZug END;
  END (* WHILE *)
  Zug := BesterZug; Wert := BesterWert;
END;
```

Und nun sehen wir uns das partielle Gegenstück an. Die Besonderheiten sind in schräger Schrift gedruckt.

```
PROCEDURE Minimax (Tiefe,Spieler: INTEGER;
                   VAR Wert: INTEGER; VAR Zug: ...);
(* Tiefe = Suchtiefe, Spieler = 0/1, Wert = Feldbewertung *)
VAR TestZug,BesterZug: ...; BesterWert: INTEGER;
    ZugListe: ...; WertListe: ...;
BEGIN
  WHILE "Es gibt noch mögliche Züge" DO
  BEGIN
    TestZug := "möglicher Zug";
    "Ziehe den Zug TestZug";
    ZugListe[i] := TestZug; WertListe[i] := "Bewerte das aktuelle Feld";
    "Nimm den Zug TestZug zurück";
    i := i+1;
  END (* WHILE *)
  "Lösche aus der ZugListe die Züge aus, die gestrichen werden sollen.
   Die WertListe dient als Entscheidungskriterium";
  WHILE "ZugListe nicht leer" DO
```

```
  BEGIN
    TestZug := ZugListe[i];
    "Ziehe den Zug TestZug";
    IF Tiefe > 1 THEN
      Minimax (Tiefe-1, 1-Spieler, Wert, Zug);    (* Rekursiver Aufruf *)
      (* Tiefe nur noch eins weniger, anderer Spieler ist an der Reihe *)
    ELSE Wert := "Bewerte das aktuelle Feld";
    "Nimm den Zug TestZug zurück";
    IF Wert > BesterWert THEN
    BEGIN BesterWert := Wert; BesterZug := TestZug END;
  END (* WHILE *)
  Zug := BesterZug; Wert := BesterWert;
END;
```

Quintessenz

Heuristik ist in der Spieltheorie ein sehr wichtiges Thema. Sie lässt sich überall dort einsetzen, wo spezifische Algorithmen fehlen. In der professionellen Softwareentwicklung wird sie in den Gebieten der statischen Feldbewertung rege benutzt. Von der partiellen Suche ist man wieder ein bisschen weggekommen, weil dadurch das trickreiche Spielen verlorengeht. Vielleicht sähe dies etwas anders aus, wenn ein gute Heuristik für die Auswahl bekannt wäre. Vielleicht gibt es sogar Spiele, wo diese Auswahl relativ einfach ist. Auf alle Fälle muss eines gelten: Probieren geht über studieren. Zumindest in der Heuristik.

SchülerInnen-Lernkontrolle

(Serie A)

Hier folgen nun drei Testfragen. Deine Antworten auf diese Fragen werden nicht bewertet. Du brauchst auch nichts abzugeben. Diese Fragen sollen Dir helfen zu erkennen, ob Du den Stoff begriffen hast. Du solltest mindestens zwei der drei folgenden Fragen richtig beantworten können. Falls Du dies nicht schaffst, solltest Du die entsprechenden Stellen im Kapitel noch einmal durcharbeiten. In diesem Fall kannst Du diese Fragen ja nachher noch einmal angehen.

Verwende zur Bearbeitung keine Unterlagen!

Und noch etwas: Niemand wird kontrollieren, dass Du diesen Test auch wirklich machst und bestehst. Doch Deine MitschülerInnen haben ein Anrecht auf eineN kompetenteN Experten/-in.

Aufgabe 1
Wie gross ist die Suchbreite des Bauernschachs gleich zu Spielbeginn?
Wie gross schätzt Du die durchschnittliche Suchbreite des Bauernschachs? Erkläre, warum Du nicht weniger und warum Du nicht mehr schätzt.

Aufgabe 2
Die Formel zur Berechnung der Anzahl Feldbewertungen im partiellen Suchbaum lautet

$$\frac{n^d - 1}{n - 1} \cdot b$$

wobei d=Suchtiefe, b=Suchbreite, n=Anzahl Züge, die weiterverfolgt werden.
Im **Bauernschach** ist b=3. Ausserdem wählen wir d=3 und n=2. Berechne anhand der Formel, wieviele Felder bewertet werden müssen, damit der Computer einen Zug spielen kann. Zeichne den Suchbaum auf und kontrolliere, ob der Wert der Formel stimmt.

Aufgabe 3
Gegeben sind die folgenden Feldbewertungsfunktionen für ein **Vier Gewinnt**:
(i) Anzahl weisse Steine minus Anzahl schwarze Steine.
(ii) Jeder Stein gibt so viele Punkte, wie er gleichfarbige Nachbarn hat. Weisse Steine geben einen positiven, schwarze Steine einen negativen Beitrag an die Gesamt-punktzahl.
(iii) Anzahl weisse Dreierreihen minus Anzahl schwarze Dreierreihen.
(iv) Jede Reihe gibt so viele Punkte, wie sie lang ist. Schwarze Reihen geben einen positiven, weisse Reihen einen negativen Beitrag an die Gesamtsumme.
Gib zu jeder aufgezählten Bewertungsfunktion einen Grund an, warum Du diese nicht verwenden würdest. Suche bitte zu jeder Funktion einen anderen Grund!

Lösungen

(Serie A)

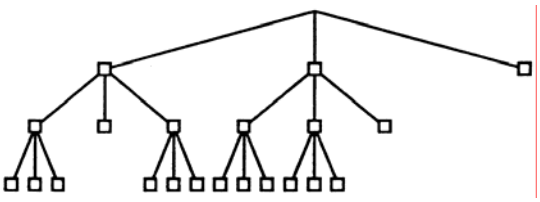
Aufgabe 1

Im ersten Zug hat Weiss drei mögliche Züge. Also beträgt die Suchbreite gleich zu Spielbeginn drei.

Die durchschnittliche Suchbreite ist etwa drei. Es gibt Stellungen, in denen mehr Züge zur Verfügung stehen, aber besonders nachdem Figuren geschlagen wurden stehen meistens nicht einmal mehr drei Züge zur Auswahl.

Aufgabe 2

$$b = 3, d = 3, n = 2 \Rightarrow \frac{n^d - 1}{n - 1} \cdot b = 21$$

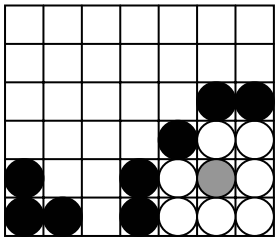


Im Suchbaum stehen 21 Knoten, die Feldbewertungsfunktion muss also für 21 Stellungen ausgewertet werden. Dies entspricht dem Ergebnis, das die Formel liefert.

Aufgabe 3

Hier zu jedem Punkt einen Nachteil:

(i) Diese Bewertungsfunktion gibt nur an, wer an der Reihe ist. Die Spieler ziehen abwechselnd, wenn Weiss am Zug ist, haben beide Spieler gleich viele Steine im Spiel, wenn Schwarz an der Reihe ist, hat Weiss einen Stein mehr.



(ii) Selbst wenn ein Stein viele gleichfarbige Nachbarn hat, kann er vollkommen blockiert sein! Er kann zum Beispiel von einem Ring gleichfarbiger Nachbarn umgeben sein, welcher wiederum von einem Ring andersfarbiger Steinen begrenzt wird. Ein solcher Stein darf keine Punkte geben, da er nicht zu einem Gewinn verhelfen kann.

(iii) Eine Dreierreihe kann zwar zum Sieg verhelfen, sie muss aber mindestens ein freies Ende haben. Sonst ist sie nichts wert.

(iv) Die Viererreihe darf keinesfalls nur vier Punkte geben, sonst werden zwei Dreierreihen einer Viererreihe bevorzugt. Die Viererreihe muss viel mehr Punkte geben.

Computerlernen

Übersicht

In diesem Kapitel über Computerlernen geht es hauptsächlich um die Frage, wie ein Computer *lernen* kann. In diesem Zusammenhang wollen wir den Begriff "Lernen" als Wissenserweiterung betrachten. Die Darstellbarkeit des Wissens interessiert uns allerdings nur am Rande. Vielmehr wollen wir zu verstehen versuchen, wie ein Computer eben dieses Wissen erweitern kann. Ziel ist es, dass der Computer aus seinen eigenen Fehlern lernt: Wenn der Computer am Ende einer Spielrunde verloren hat, dann muss er offensichtlich etwas lernen. Er soll in Zukunft nicht mehr auf die gleiche Art und Weise verlieren. Er soll lernen, welche Züge schlecht und welche gut waren.

Vorgehen

Wissenserwerb:
Studiere die vorliegenden Unterlagen sorgfältig. Wenn Du das Gefühl hast, Du hast den Stoff begriffen, so löse die Aufgaben der Schüler-Lernkontrolle ganz am Ende.
Expertenrunde:
Besprecht in der Gruppe, *was* Ihr Euren MischülerInnen mitteilen wollt. Beachtet dazu die unten angegebenen Lernziele!
Überlegt Euch, *wie* Ihr diesen Stoff den ZuhörerInnen vermittelt wollt.

Lernziele

Nachdem Du dieses Kapitel durchgearbeitet hast, weisst Du, inwiefern ein Computer lernen kann. Schlagwörter wie *Künstliche Intelligenz* oder *Wissensbasierte Systeme* kannst Du in einen Zusammenhang stellen.
Du kannst umgangssprachlich erklären, was ein Wissensbasiertes System ist und wie sich die drei wichtigsten Arten unterscheiden. Du kannst einem Aussenstehenden auch zeigen, dass dies für "normale" Spiele wie **Schach** oder **Mühle** so einfach nicht geht, indem Du ihm den benötigten Speicherplatz vorrechnest.

Material

- "Bauernschach", eine Einführung in ein vereinfachtes Schachspiel.

Computerlernen

Begriffe

Gleich zu Beginn wollen wir die fortan verwendeten Begriffe kurz einführen. Dies ist notwendig, da verschiedene Autoren die Begriffe oft verschieden verwenden.

KI ist die Abkürzung für *Künstliche Intelligenz*. Darunter fallen Forschungsgebiete wie automatisches Beweisen, Text- und Spracherkennung, Fehlertoleranz und vieles mehr. Die Methoden sind vor allem Expertensysteme, Neuronale Netze und Wissensbasierte Systeme.

Ein **Wissensbasiertes System (WBS)** ist ein Programm, das viele Beispiele von Problemen mit den Lösungen kennt. Es kann Probleme, die *gleich* sind wie ein bekanntes, lösen (durch nachschlagen). Das Programm besteht im Wesentlichen aus einer Datenbank, also aus einer Ansammlung von Daten. Die Daten sind eigentlich Regeln. Diese haben alle das Format "Wenn ... dann ...". Eine mögliche Regel im Schach wäre etwa: "Wenn König erreichbar von gegnerischer Dame dann ...". Jede einzelne Regel bewirkt nur sehr wenig. Erst in grossen Mengen werden sie leistungsstark.

In diesem Kapitel beschäftigen wir uns hauptsächlich mit Wissensbasierten Systemen und Lernenden Automaten. Wir gehen aber nicht auf die Details oder auf Optimierungsmöglichkeiten ein, sondern wollen nur das Grundprinzip vermitteln.

Was ist ein Wissensbasiertes System in der Spieltheorie?

Ein Wissensbasiertes System besteht hauptsächlich aus einer grossen Ansammlung von Regeln. Jede Regel besteht aus einer Bedingung (Wenn ...) und einer Folge (dann ...). Die Bedingung besteht aus einer "eingefrorenen" Spielstellung, also eigentlich einem Photo des Spielbretts mit allen Figuren und einem Hinweis, welcher Spieler an der Reihe ist. Die Folge ist ein Zug, der entweder gezogen oder eben gerade nicht gezogen werden soll.
Nun stellt sich natürlich die Frage, wo denn die Datenbank mit all den tollen Zügen herkommt. Des Rätsels Lösung findest Du im nächsten Abschnitt.

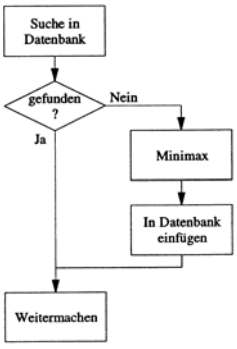
Wie kommen die Regeln ins System?

Zu Beginn ist das Wissensbasierte System leer. Es gibt zwar Programmierer, die dem System schon ein paar Regeln mit auf den Lebensweg geben, aber prinzipiell soll ein solches Programm in der Lage sein, selber Regeln zu finden. Dazu gibt es viele Möglichkeiten. Drei verbreitete Modelle wollen wir uns genauer anschauen:

- Systeme ohne Rückmeldungen
- Systeme mit negativen Rückmeldungen
- Systeme mit positiven und negativen Rückmeldungen

WBS ohne Rückmeldungen

Die einfachste Art von Wissensbasierten Systemen funktioniert ohne Rückmeldungen. Das bedeutet, dass Züge, die gespielt werden, immer ins WBS übernommen werden. Ausserdem enthält die Datenbank nur positive Regeln, also solche, die zu einer gegebenen Spielsituation den Zug angeben, der gespielt werden soll. Der Lernprozess funktioniert wie folgt:
Wichtig ist, dass ein Zug in die Datenbank eingefügt wird, falls dieser noch nicht drin ist.



Dieser Vorgang bildet das eigentliche Lernen. Ein solches Wissensbasierte System ist nicht

im eigentlichen Sinn lernfähig. Es merkt sich lediglich seine eigenen Berechnungen und führt diese so nur einmal aus.

WBS mit negativen Rückmeldungen

Diese Methode ist auch unter dem Namen *Streichverfahren* bekannt.
Zu Beginn sind in der Datenbank alle möglichen Spielstellungen mit allen möglichen Zügen gespeichert. Wenn der Computer am Zug ist, so sucht er die gegebene Stellungen in seiner Datenbank. Dort findet er, welche Züge in dieser Situation erlaubt sind. Daraus wählt er mit einem beliebigen Verfahren (meistens zufällig) einen Zug.
Wenn der Computer ein Spiel verliert, so kann er daraus schliessen, dass sein letzter Zug schlecht war. Dieser führte entweder direkt zu seiner Niederlage, oder er eröffnete seinem Gegner die Möglichkeit zu gewinnen. Also wird dieser Zug aus der Datenbank gestrichen und kann so in Zukunft nicht mehr gewählt werden.
Über den zweitletzten Zug lässt sich hingegen keinerlei Aussagen machen. Vielleicht hätte ja der Gegner im letzten Zug noch das ganze Spiel zu seinen Gunsten kehren können. Also belassen wir alle anderen Züge in der Datenbank.
Wenn der Computer während des Spiels in die Situation kommt, keinen einzigen Zug zu dieser Stellung in seiner Datenbank zu haben, so bedeutet dies das folgende: Jeder mögliche Folgezug wurde schon einmal ausprobiert und hat zur Niederlage geführt. Also kann[5] mit dieser Stellung der Gegner gewinnen, egal wie gut der Computer auch spielen mag. Folglich kann diese Stellung auch bereits als Niederlage des Computers gewertet werden. Also muss der Computer bemüht sein, zukünftig nicht mehr in diese missliche Lage zu kommen. Dazu wird der letzte Zug des Computers aus der Datenbank entfernt.

Wenn dieses Spielchen lange genug getrieben wird, spielt der Computer perfekt. Das heisst, wenn der Computer überhaupt gewinnen kann, dann gewinnt er auch. Definitiv.

Und wo liegt der Hase begraben?
Die Geschichte hat einen Haken. Wie lange ist "lange genug"?
Nehmen wir als Beispiel ein normales, siebenspaltiges **Vier Gewinnt** mit runterfallenden Steinen. Berechnen wir die Anzahl Stellungen dieses doch noch recht einfachen Spieles:

$$\text{Anzahl Stellungen} \approx \underbrace{\left(\sum_{i=0}^{7} \underbrace{2^{i}}_{Schwarz\,/\,Weiss} \right)}_{i=\text{Höhe dieser Spalte}}^{\overset{\text{Anzahl Spalten}}{7}} = \left(2^{7+1}-1\right)^{7} \approx 7 \cdot 10^{16}$$

Die Anzahl der Stellungen, die gespeichert werden müssen, ist enorm. Damit könnte man mehrere Millionen Festplatten à je 1 Gigabyte (= 1000 Megabyte) füllen. Oder bildlich ausgedrückt: Dies entspricht einem Stapel vollbedruckter A4-Blätter bis zum Mond. Beidseitig bedruckt, versteht sich.

Auch die Lernzeit ist astronomisch gross. Da in jeder Partie höchstens ein Zug aus der Datenbank entfernt wird, dürfte es etwas lange dauern, bis das Wissensbasierte System trainiert ist.

Aber dieses Modell ist trotzdem wichtig. Es beweist nämlich, dass es eine optimale Strategie gibt. Der einzige Grund, dass ein solches System nicht funktioniert, ist die Beschränktheit der Ressourcen (Speicher und Geschwindigkeit). Natürlich kann man sich darüber streiten, wie

[5] Der Gegner sollte schon die richtigen Züge auswählen..

sinnvoll Modelle sind, die nie in Programmen verwendet werden können. Aber die theoretische Erkenntnis ist gross.

WBS mit positiven und negativen Rückmeldungen

Ein drittes Modell arbeitet sowohl mit positiven als auch mit negativen Rückmeldungen. Dies bedeutet, dass die Datenbank sowohl bei einem Sieg als auch bei einer Niederlage verbessert wird. Bei einer Niederlage wir die Regel gespeichert, dass in der Spielstellung vor dem letzten Zug eben dieser letzte Zug nicht gespielt werden soll. Dieser hatte nämlich eine Niederlage zur Folge. Im Siegesfalle werden alle in dieser Partie gespielten Züge mit den dazugehörigen Stellungen gespeichert. Offenbar waren dies gute Züge, denn der Gegner hat die Partie verloren.
Zu Beginn ist die Datenbank leer. Wenn der Computer am Zug ist, so sucht er die aktuelle Stellung in der Datenbank. Wenn er sie nicht findet, spielt er zufällig (oder nach einem beliebigen System) einen möglichen Zug. Wenn er zu einer Stellung gute Züge findet, spielt er einen davon. Findet er schlechte Züge, spielt er einen anderen.

Dieses Modell eignet sich nicht für theoretische Erkenntnisse. Auch nach unendlicher Zeit spielt der Computer nicht perfekt. Die guten Züge brauchen nämlich nicht zum Sieg zu führen. Vielleicht macht der Gegner immer den selben Fehler, so dass ein bestimmter Zug zum Sieg der Partie führt. Würde der Gegner diesen Fehler nicht machen, könnte er gewinnen. Ein positiver Zug in der Datenbank sagt also nur aus, dass mit diesem Zug schon einmal eine Partie gewonnen wurde. Aber dies bedeutet nicht, dass ein nächstes Mal bei gleicher Stellung der gleiche Zug wieder einen Sieg zur Folge hat.

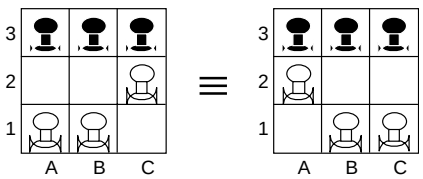
Bauernschach

Wir wollen an dieser Stelle ein Beispiel betrachten, wie ein einfaches Spiel mit Hilfe des Streichverfahrens (WBS mit negativen Rückmeldungen) gespielt werden kann.

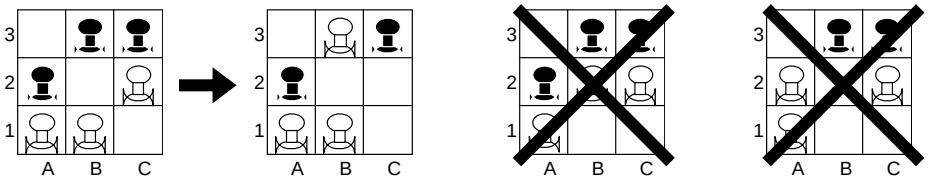
Studiere dazu bitte das Dokument "Bauernschach".

Die Anzahl der möglichen Stellungen im **Bauernschach** beträgt etwa zweihundert. Dies ist eine für einen Computer absolut problemlose Grösse. Wir wollen hier das **Bauernschach** aber ohne Computer analysieren. Mit den folgenden beiden Annahmen können wir die Anzahl der Stellungen noch weiter einschränken!

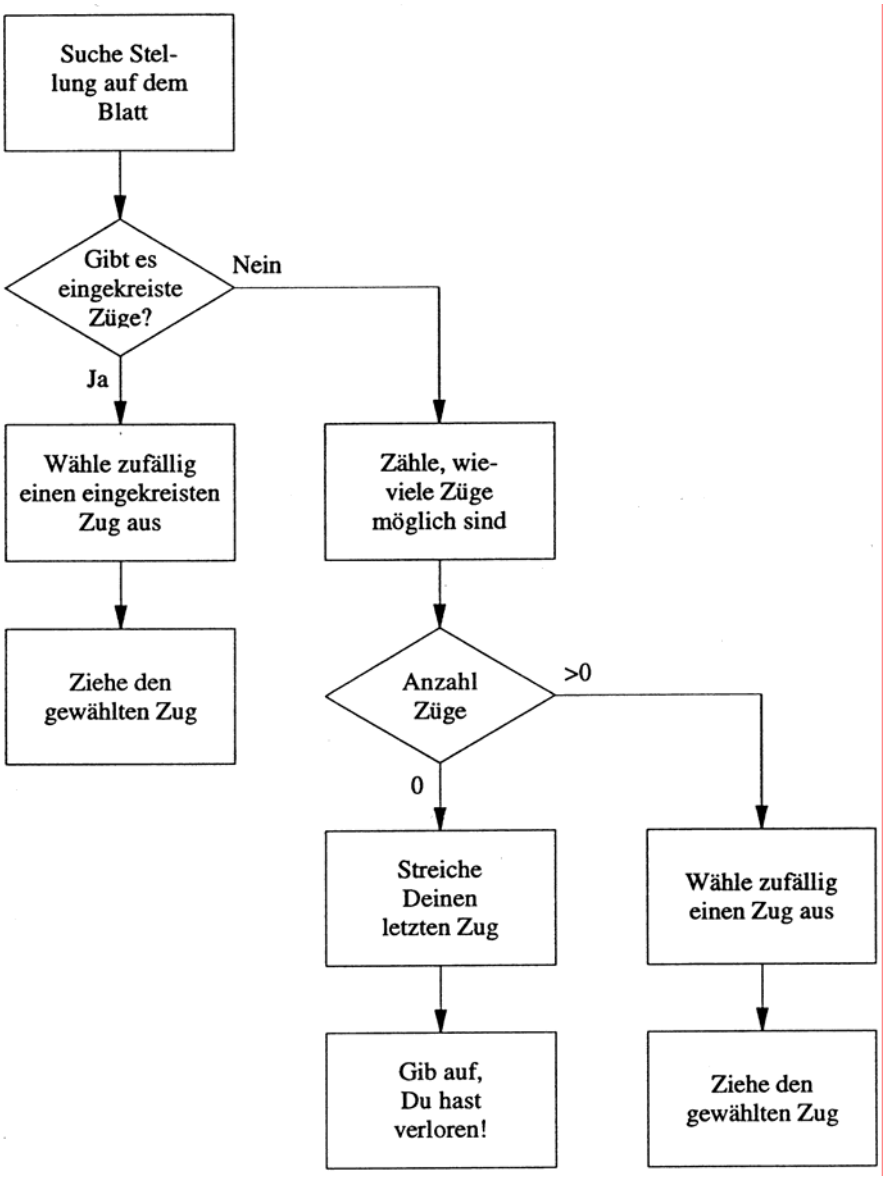
- Wir eliminieren Spiegelungen:



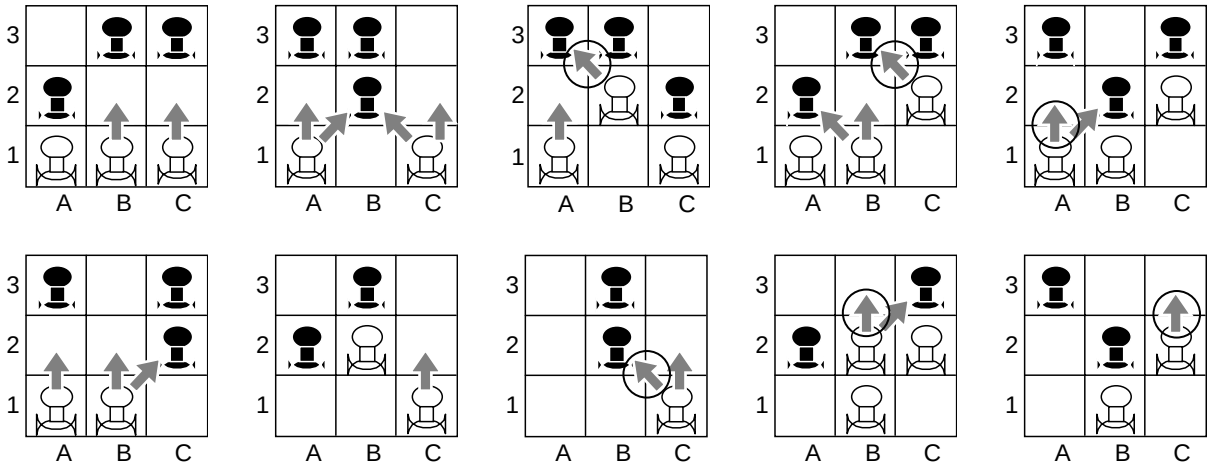
- Wenn der ziehende Spieler einen direkt zum Sieg führenden Zug spielen kann, so spielt er diesen auch. Die anderen Züge werden in diesem Fall nicht mehr weiterverfolgt.



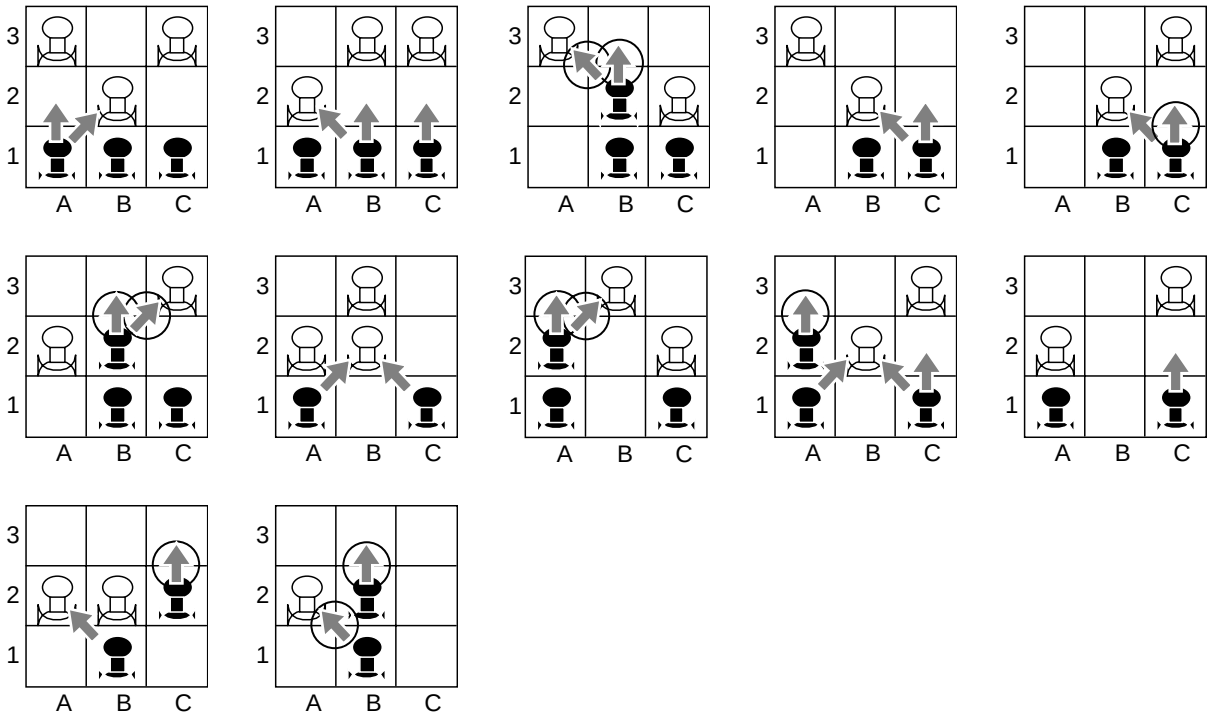
Aufgabe:
Unten findest Du eine Liste aller möglichen Spielstellungen, einmal aus der Sicht von Weiss und einmal aus der Sicht von Schwarz. Spiele mit Deinem/-r Nachbarn/-in einige Spiele durch. Wenn Du an der Reihe bist, dann suchst Du die Stellung auf Deinem Blatt. Wenn Du die Stellung gefunden hast, aber es gibt keine möglichen Züge mehr, dann streiche Deinen letzten gemachten Zug auf dem Blatt durch. Wenn einer der möglichen Züge in dieser Stellung eingekreist ist, dann ist dies ein Zug, der direkt zum Sieg führt. Nach der Abmachung von oben musst Du diesen Zug spielen. Falls keine Züge eingekreist sind, wählst Du zufällig einen der möglichen Züge auf dem Blatt aus und spielst ihn.



Hier folgen nun die möglichen Spielstellungen aus der Sicht von Weiss:



Und hier die möglichen Spielstellungen aus der Sicht von Schwarz:



SchülerInnen-Lernkontrolle

(Serie A)

Hier folgen nun drei Testfragen. Deine Antworten auf diese Fragen werden nicht bewertet. Du brauchst auch nichts abzugeben. Diese Fragen sollen Dir helfen zu erkennen, ob Du den Stoff begriffen hast. Du solltest mindestens zwei der drei folgenden Fragen richtig beantworten können. Falls Du dies nicht schaffst, solltest Du die entsprechenden Stellen im Kapitel noch einmal durcharbeiten. In diesem Fall kannst Du diese Fragen ja nachher noch einmal angehen.

Verwende zur Bearbeitung keine Unterlagen!

Und noch etwas: Niemand wird kontrollieren, dass Du diesen Test auch wirklich machst und bestehst. Doch Deine MitschülerInnen haben ein Anrecht auf eineN kompetenteN Experten/-in.

Aufgabe 1
Wenn Du im **Bauernschach** auswählen darfst, mit welcher Farbe Du spielen willst: Wie entscheidest Du Dich?

Aufgabe 2
Warum werden Computerspiele heute nicht alle in Form Wissensbasierter Systeme auf den Markt gebracht?

Aufgabe 3
Wir haben gesehen, dass es im **Vier Gewinnt** ungefähr $7 \cdot 10^{16}$ Spielstellungen gibt. Für das Streichverfahren müssten anfangs zu jeder Stellung 7 Züge gespeichert sein.
Nehmen wir vereinfacht an, jede Schachpartie dauert exakt 40 Züge, und in jeder Spielstellung gibt es genau 20 mögliche Züge. Ausserdem ignorieren wir, dass auf verschiedenen Wegen die gleiche Stellung erreicht werden kann.
Wieviele Stellungen müssten im Streichverfahren für dieses idealisierte Schach zu Beginn gespeichert sein?

Lösungen

(Serie A)

Aufgabe 1
Wenn der Spieler mit den schwarzen Figuren richtig spielt, dann gewinnt er immer. Ich würde mich deshalb für die schwarzen Figuren entscheiden.

Aufgabe 2
Weil die heutigen Computer nicht genug (Festplatten-) Speicher haben, um zu jeder möglichen Stellung den besten möglichen Zug zu wissen.

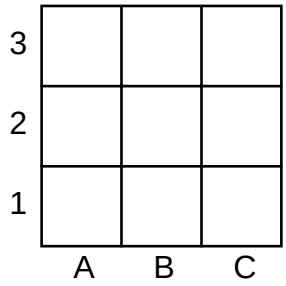
Aufgabe 3
Anzahl mögliche Stellungen nach einem Zug: 20
Anzahl mögliche Stellungen nach zwei Zügen: 20^2
Anzahl mögliche Stellungen nach drei Zügen: 20^3
Anzahl mögliche Stellungen nach 40 Zügen: $20^{40} \approx 10^{52}$

Bauernschach

Bauernschach ist eine Vereinfachung des normalen Schachspiels. Die Idee dazu stammt von Helmut Seul und Martin Gardner.

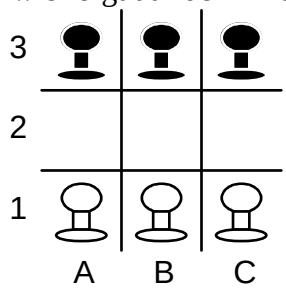
Das Spielbrett

Das Spielbrett besteht aus drei mal drei Feldern.



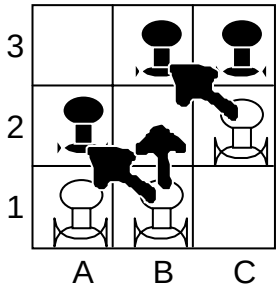
Die Figuren

Die einzigen Figuren des **Bauernschachs** sind Bauern! Beide Spieler besitzen bei Spielbeginn drei Bauern. Diese sind wie folgt auf dem Brett aufgestellt:



Wie wird gezogen?

Der Spieler mit den weissen Figuren zieht nach oben, der Spieler mit den schwarzen Figuren nach unten. In einem Zug darf ein Spieler nur einen Bauern und diesen nur um ein Feld bewegen.
In einem normalen Zug wird der Bauer ein Feld geradeaus nach vorne gestossen. Das Zielfeld muss leer sein!
In einem schlagenden Zug wird der Bauer entweder schräg nach links vorne oder nach rechts vorne gezogen. Das Zielfeld muss mit einer gegnerischen Figur besetzt sein. Diese Figur wird vom Spielfeld genommen.
Im folgenden Bild sind alle möglichen Züge des weissen Spielers angegeben. Die Spielstellung ist rein zufällig!



Der Spielablauf

Die beiden Spieler spielen abwechselnd einen Zug. Der Spieler mit den weissen Figuren beginnt.

Wer gewinnt?

Der Spieler, der als erstes eine Figur auf die letzte Zeile (Zeile 3 für weiss, Zeile 1 für schwarz) des Spielbretts zieht, gewinnt.
Wenn ein Spieler an der Reihe ist und nicht ziehen kann (weil er blockiert ist oder weil er keine Figuren mehr hat), dann gewinnt der andere Spieler.

In der oben gezeichneten Stellung wird also der Spieler mit den weissen Figuren sicherlich c2b3 spielen, da er mit diesem Zug gerade gewinnt!

Vollständiger Spielbaum

Das folgende Bild ist nur für den Lehrer bestimmt. Es zeigt den vollständigen Spielbaum von Bauernschach.