Learning Search Control

Knowledge Engineering und Lernen in Spielen

Vortrag von Simone Daum

Überblick

In Spielen wie Schach und Othello stützt man sich stark auf Brute-Force-Ansätze wie α/β-Algorithmen und versucht, diese durch Parametrisierung effizienter zu gestalten.
Beispiele, die im folgenden betrachtet werden:

- Move Ordering
- Search Extension Parameter
  zur Kontrolle der Suchtiefe

... in der Domäne Schach

Move Ordering

Move Ordering
Die „Chessmap“-Heuristik

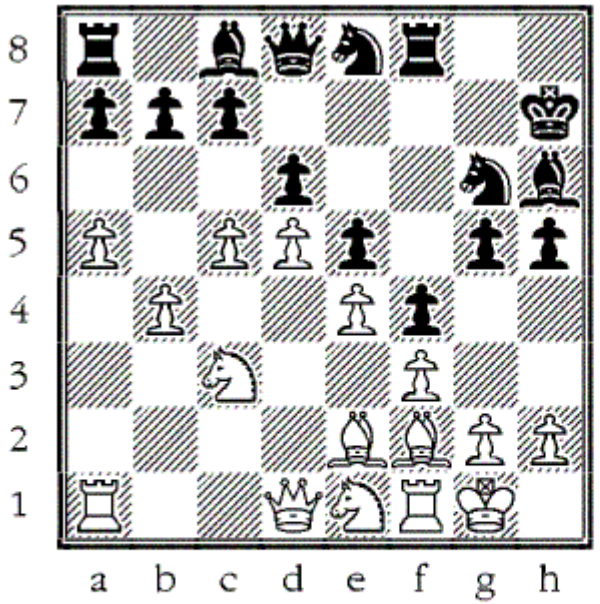
(1999, von Kieran R.C. Greer u. a.)

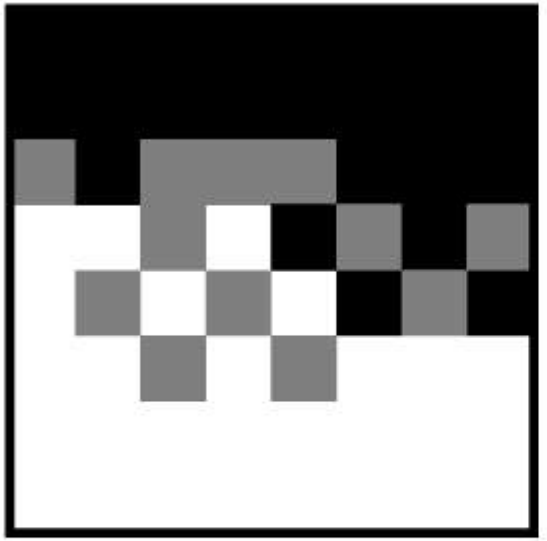
Hypothese:

Es gibt eine starke Abhängigkeit zwischen den Feldern, die von einem Spieler kontrolliert werden, und den Feldern, die von seinem Zug beeinflußt werden!

Die Schachkarte

Eine Karte, die anzeigt, welche Felder von welchem Spieler kontrolliert werden:





King's Indian Defence. Korchnoi-Kasparov, Dresden 1992

Kontrolle eines Feldes

(im folgenden immer aus der Sicht von Weiß)

Feld besetzt:

- Figur wird weder bedroht noch verteidigt -> neutrales Feld
- Figur wird verteidigt, aber nicht bedroht
  -> Kontrolle durch Weiß
- Figur wird von Schwarz bedroht, aber Schwarz würde beim Schlagen Material verlieren
  -> Kontrolle durch Weiß
- Figur wird von Schwarz bedroht, aber Schwarz würde Material gewinnen
  -> Kontrolle durch Schwarz
- Figur wird von Schwarz bedroht, aber keiner würde Material gewinnen -> neutrales Feld

Kontrolle eines Feldes (2)

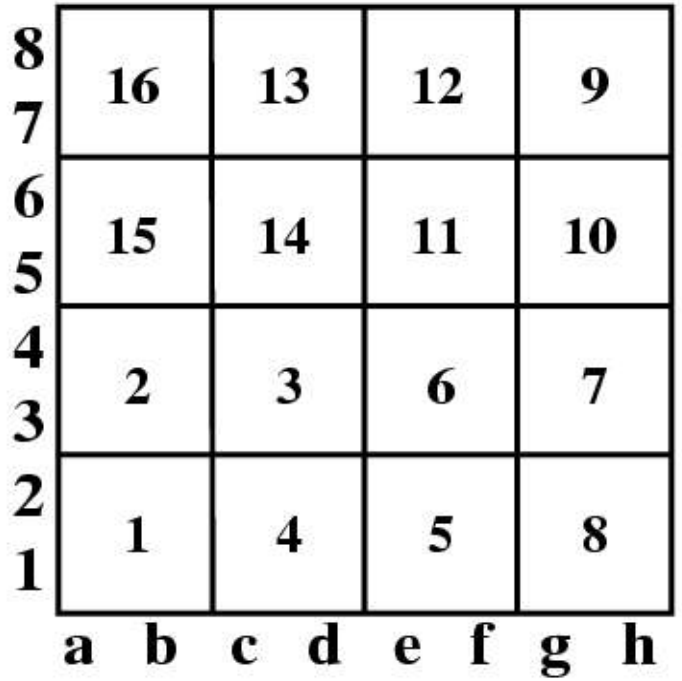
(im folgenden immer aus der Sicht von Weiß)

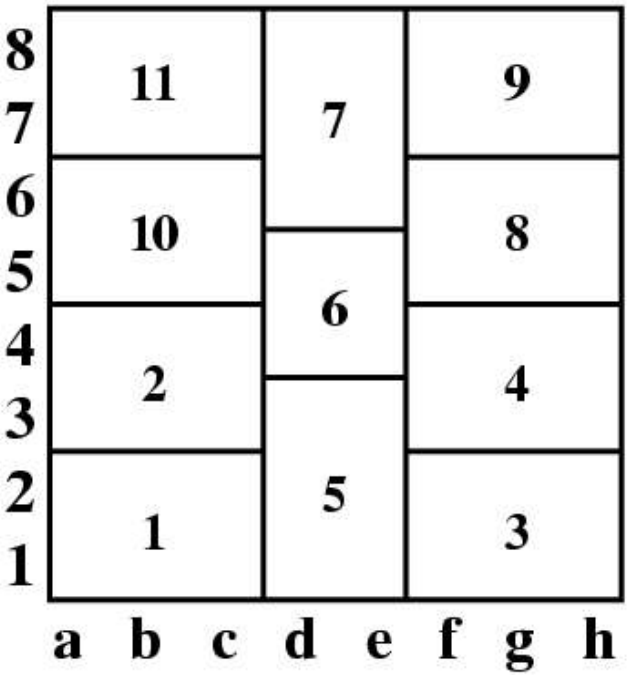
Feld nicht besetzt:

- Kein Spieler kann auf das Feld ziehen
  -> neutrales Feld
- Weiß kann ohne Materialverlust eine wertvollere Figur auf das Feld ziehen als Schwarz
  -> Kontrolle durch Weiß
- Schwarz kann ohne Materialverlust eine wertvollere Figur auf das Feld ziehen als Weiß
  -> Kontrolle durch Schwarz
- Beide Seiten könnten mit gleich wertvoller Figur auf das Feld ziehen
  -> neutrales Feld

Einfluß eines Zuges

Schachbrett wird in Sektoren aufgeteilt, die von unterschiedlicher Größe und Form sein können:





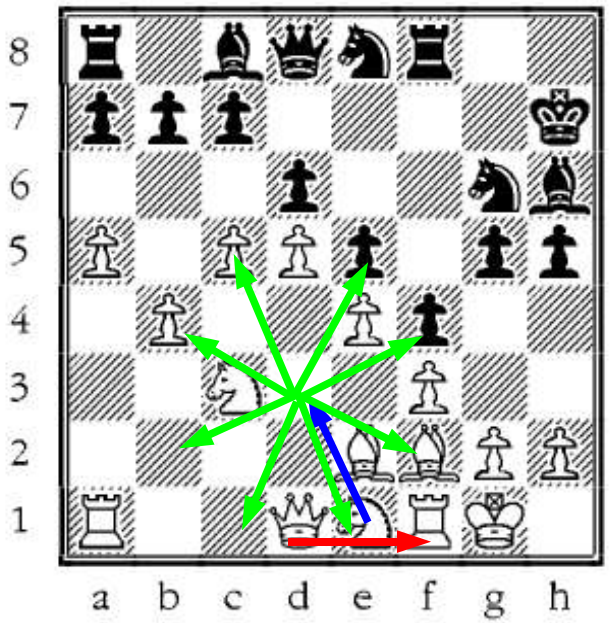
Einfluß eines Zuges (2)

Ein Sektor wird von einem Zug beeinflußt, falls:

- die Figur in den Sektor zieht
- ein Feld im Sektor nach dem Zug von der Figur bedroht/gedeckt wird
- eine Figur nun ein Feld im Sektor bedroht/deckt, nachdem sie zuvor von der ziehenden Figur blockiert wurde

Einfluß eines Zuges (3)

Beispiel:



Se1-d3
beeinflußt folgende Felder:

- d3
- c1, e1, f2, f4, e5, c5, b4, b2
- f1 (Dame deckt Turm)

Move Ordering

Einfach Regel:

Solche Züge, die wichtige Sektoren beeinflußen, werden vor Zügen betrachtet, die nur untergeordnete Sektoren beeinflußen!!!

Aber...

Welche Sektoren sind für den Spieler wichtig?

Das Festzulegen ist eigentliches Ziel der Heuristik!

Relevanz eines Sektors

1. Methode:

Aufsteigende Ordnung der Sektoren nach Anzahl der Felder, die vom Spieler kontrolliert werden.

Bei gleicher Anzahl an Feldern wird eine feste Rangfolge eingehalten.

Relevanz eines Sektors (2)

Festgelegte Rangfolge bei gleicher Anzahl kontrollierter Felder:

16	**9**	**7**	**13**
11	**3**	**4**	**8**
15	**1**	**2**	**10**
14	**5**	**6**	**12**

Relevanz eines Sektors (3)

2. Methode:

Verwenden eines neuronalen Netzes

Input:

- 64-elementiger Vektor (entspr. den Feldern)
- „1“/ „-1“ ~ Kontrolle Weiß/Schwarz, „0“ ~ neutral

Output:

- für jeden Sektor reeller Wert, der als Relevanz des Sektors interpretiert werden kann

Erste Testergebnisse

16 Sektoren à 4 Felder

	„Felder zählen“	Neuronales Netz
Höchstwertiger Sektor vom Zug beeinflußt	44,1%	60,4%
2. höchster Sektor	20,3%	17,5%
3. höchster Sektor	10%	6,9%
Total	74,4%	84,8%

Erste Testergebnisse

64 Sektoren à 1 Feld

	Neuronales Netz
Höchstwertiger Sektor vom Zug beeinflußt	33,2%
2. höchster Sektor	16,2%
3. höchster Sektor	11,4%
Total für 4 Felder	67%
Total für 12 Felder	89,5%

Erste Testergebnisse

Ein Experiment:

Input	Wichtigste Sektoren
64x „+1“	d6, e5, d7, f7, h8
64x „0“	e1, f1, d4, f4, d5
64x „-1“	f1, e4, d3, e1, d2

Verbesserung des Move Ordering

Manchmal ist es unzureichend, nur die Ordnung der Heuristik zu betrachten, z.B.

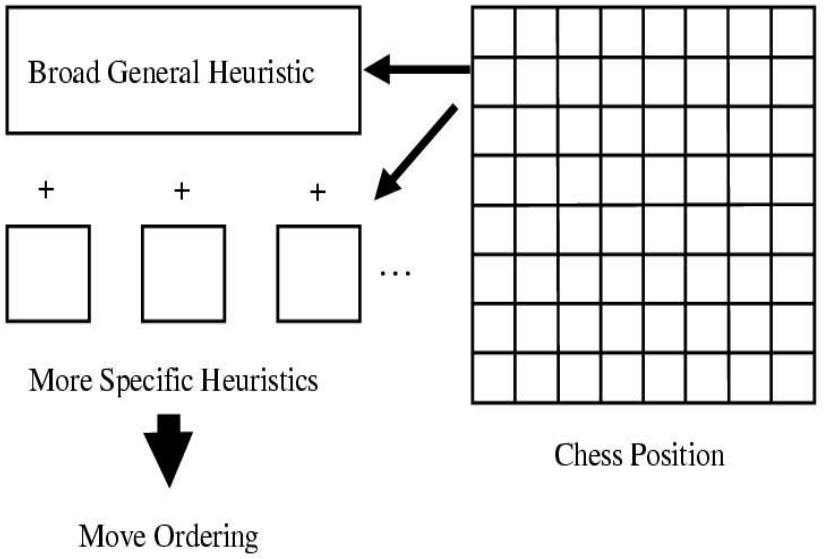
- wenn ein Zugzwang besteht
- um Züge des Königs und von Freibauern stärker zu bewerten

-> etwas Taktik hinzufügen

Unterteilen der Züge in

- sichere/unsichere Züge
- erzwungene, schlagende oder andere Züge

Verbesserung des Move Ordering (2)



Ordnung:

- Sichere, schlagende Züge
- Sichere, erzwungene Züge
- Sichere, andere Züge
- Unsichere, schlagende Züge
- Unsichere, erzwungene Züge
- Unsichere, andere Züge

Test des kompletten Move Ordering

Position	Random move ordering	Move ordering method, sectors ordered 1-64	Move ordering method, Chessmaps Heuristic	Move ordering method, Chessmaps Heuristic, store current top 2 sectors	Random move ordering, Killer Heuristic	Move ordering method, Chessmaps Heuristic, Killer Heuristic
1	741345	219924	251189	233082	548161	228893
2	858386	315893	235632	143092	486956	225618
3	4562675	290708	382342	244796	3740171	368521
4	566702	167329	284439	134303	516564	286691
5	2449130	250308	217139	182288	1437544	180930
6	970292	453101	222192	239094	943341	212809
7	1254551	258464	202441	175825	1061536	176119
8	992628	378333	180617	220170	684095	149774
9	391606	135790	217612	225958	340800	230573
10	276724	1290357	213492	142586	248207	192084
Total	**13064039**	**3760207**	**2407095**	**1941194**	**10007375**	**2252012**
%	**100**	**29**	**18.5**	**15**	**76.5**	**17**

Search Extension Parameter

Search Extension Parameter

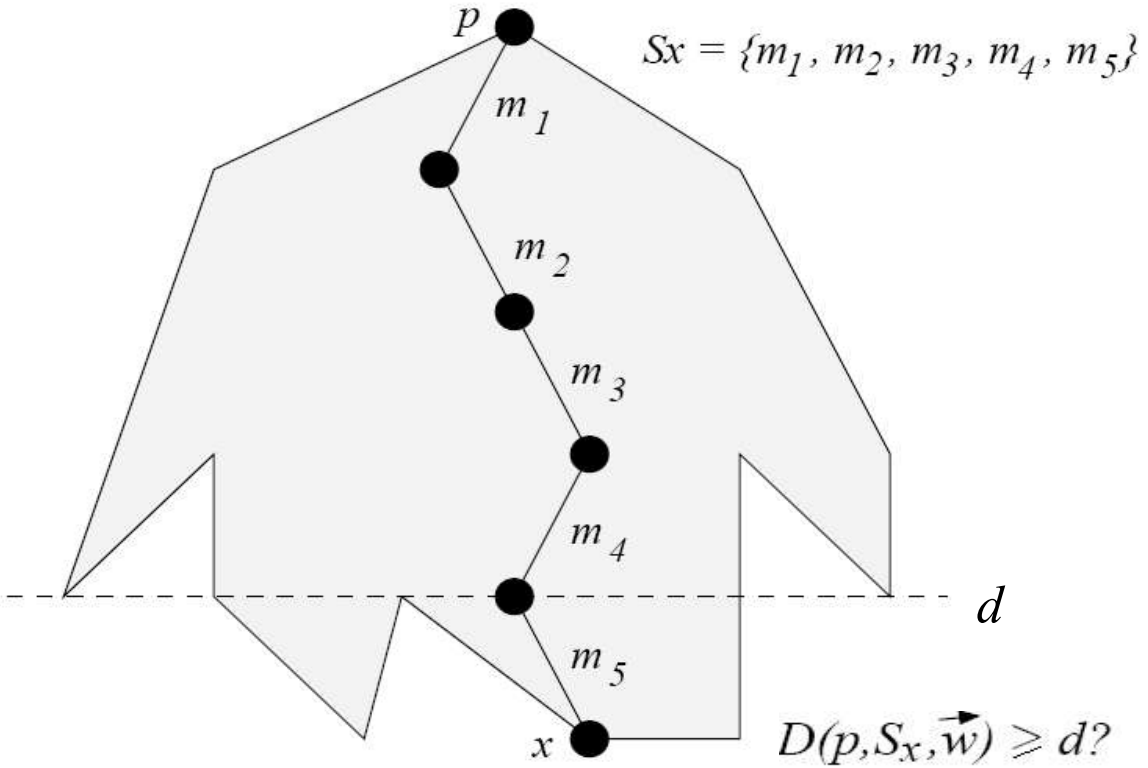
Prinzipiell werden bei einer α/β-Suche immer alle Pfade auf die gleiche Länge expandiert, aber in einer Position können einige Züge 'vielversprechender' für eine tiefere Suche erscheinen als andere.

-> Einführen von Extensionsparametern, welche die Suchtiefe steuern

Aufgabe: Lerne, diese Parameter optimal einzustellen!

Extensionsstrategie

'Verpacken' der Parameter in eine Funktion D, die man auf jeden Knoten des Spielbaums anwenden kann:



Weitere Expansion, falls

D(p,Sx,w) < d

Pfadlänge ≠ Pfadtiefe !!!

Vorauss.: D monoton steigend

Lernen der Gewichte w
- ein Minimierungsproblem

Für welche Einstellung der Gewichte w müssen am wenigsten Knoten expandiert werden, um einen gegebenen Lösungspfad zu finden?

- > Einführen einer Kostenfunktion C(p,S,w), die die Anzahl der expandiert Knoten angibt

Ziel:

$$min \sum_{(p_t,S_t)\in T} C(p_t, S_t, \vec{w})$$

... für eine Trainingsmenge T

Der Lernalgorithmus - Gradientenabstieg

```
for i= 1,N do     w[i] = 1.0  end for
while not terminate do
    nodes = 0
    for i=1,N do Δnodes[i] = 0  end for
    for all (p,S) in T do
        nodes = nodes + C(p,S,w)
        for i=1,N do
            Δnodes[i] = Δnodes[i] + ∂C(p,S,w)/∂w[i]
        end for
    end for
    for i= 1,N do
        w[i] = w[i] – μ Δw_max (Δnodes[i]/nodes)
    end for
    μ = Decrease(μ)
end while
```

Init. der Gewichte

Anzahl aller Knoten

Änderung der Knotenanz.

Änderung der Knoten-anzahl, bei Änderung eines Gewichtes

Anpassung der Gewichte

Anpassung der Lernrate

Die Kostenfunktion

Unterscheidung, ob der Lernalgorithmus zum

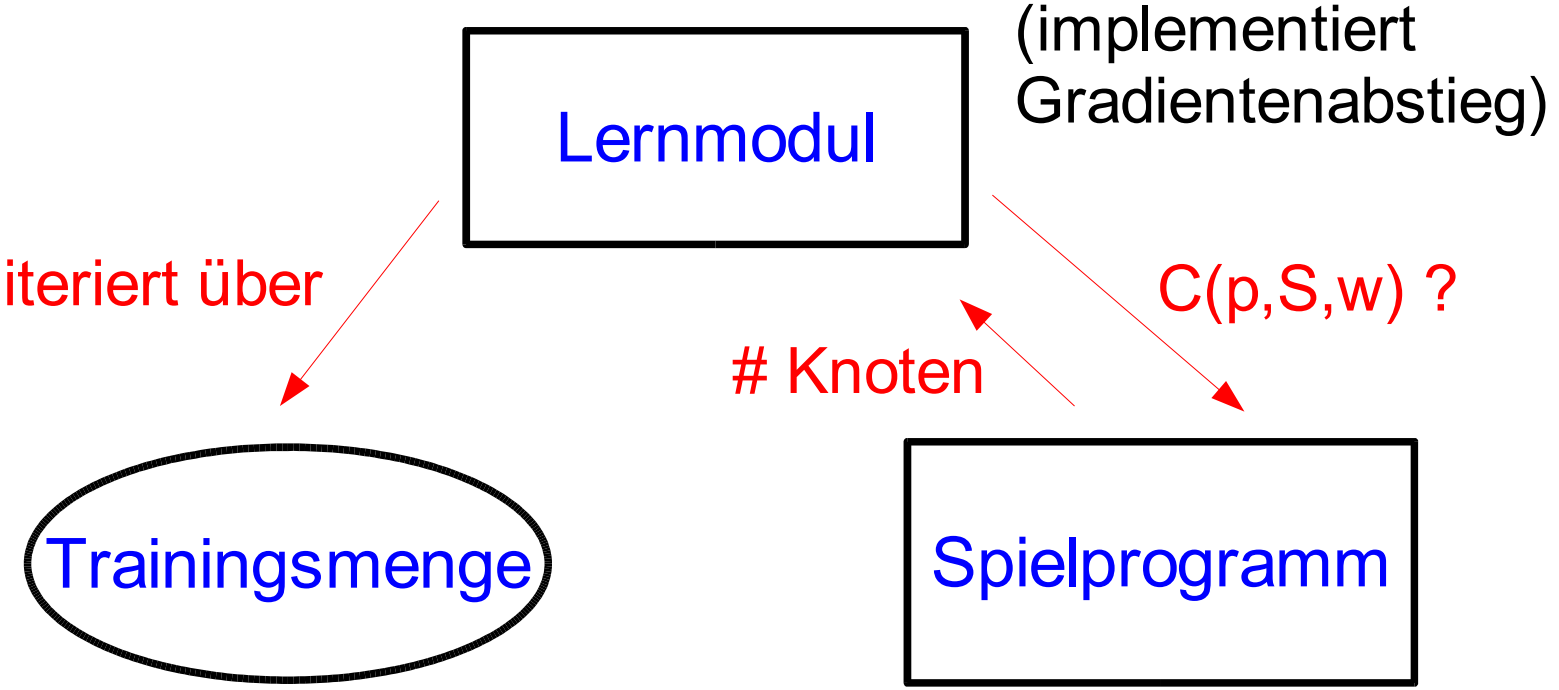
- online- oder
- offline-Lernen

eingesetzt werden soll.

-> Verschiedene Kostenmodelle

Feststellung:
Für die Kostenfunktion ist es nicht notwendig, eine stetige Funktion anzugeben, sondern es genügt, für jeden vom Algorithmus betrachteten Punkt im Suchraum einen konkreten Wert zurückzugeben!

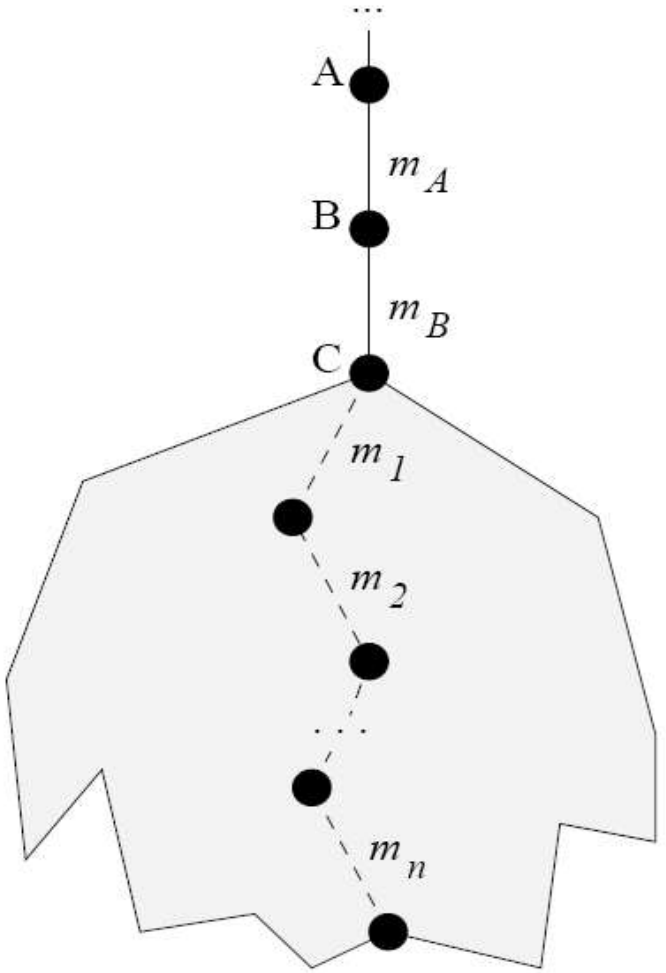
Kostenmodell zum 'offline'-Lernen



'online'-Lernen

Der Algorithmus lernt nun nicht mehr auf einer Trainingsmenge, sondern aus besonderen Situationen, die während des Spiels auftreten, sog. kritischen Situationen.

Kritische Situationen



eval(C) < eval(A) – τ

-> in A muß ein Fehler unterlaufen sein!

-> A ist eine kritische Situation

-> Benutze (A,(mA, mB, m1,.., mn)) als Trainingsbeispiel!

Kostenmodell zum 'online'-Lernen

Starten einer Tiefensuche (wie beim 'offline'-Lernen), um die Kosten zu ermitteln, ist während eines Spieles zu teuer!

-> Abschätzen der Anzahl der expandierten Knoten mit folgender Formel:

$$C(p, S, \vec{w}) = B(p, \vec{w})^{D(p,S,\vec{w})}$$

?

$$\frac{\partial C(p, S, \vec{w})}{\partial w_i} =$$

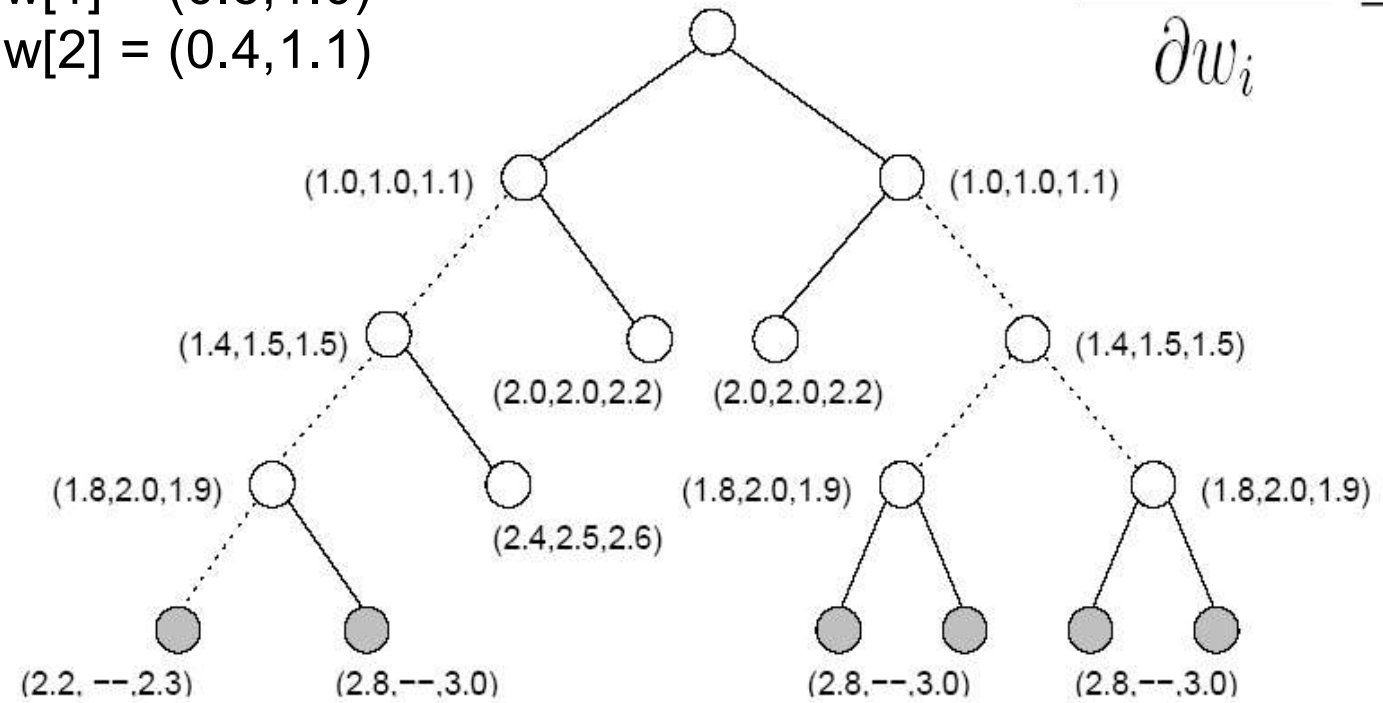
$$C(p,S, \vec{w})\left(\frac{D(p, S, \vec{w})}{B(p, \vec{w})}\frac{\partial B(p, \vec{w})}{\partial w_i} + \ln(B(p, \vec{w}))\frac{\partial D(p, S, \vec{w})}{\partial w_i}\right)$$

Wachstumsrate B(p,w) und seine partiellen Ableitungen

Annahme: B ist proportional zur Suchtiefe d

w(...,__) = (0.4,1.0)
w[1] = (0.5,1.0)
w[2] = (0.4,1.1)

$$\frac{\partial B(p,\vec{w})}{\partial w_i} = \frac{(B(p,\vec{w}_i) - B(p,\vec{w}))}{\delta}$$



B(p,w)^2 = 17
-> B(p,w) = 4,123

B(p,w[1])^2 = 11
-> B(p,w[1]) = 3,317

B(p,w[2]) = B(p,w)

'offline'- und 'online'-Lernen im Test

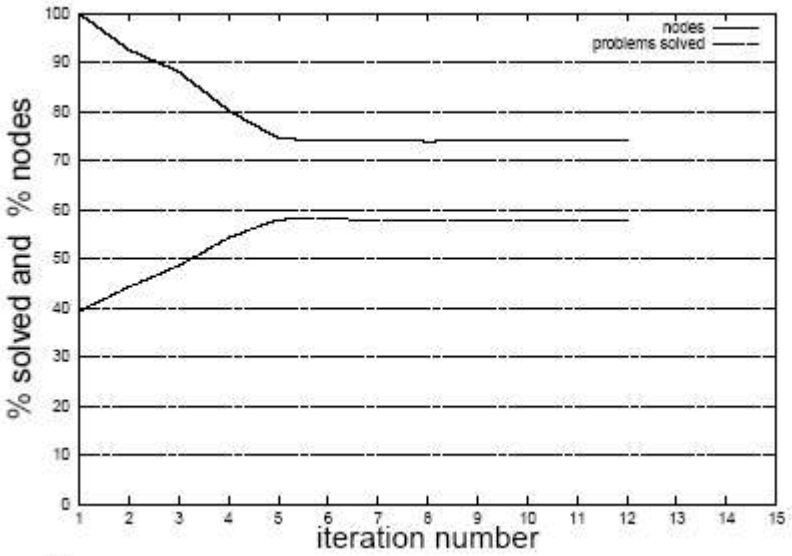
Als Spielprogramm wird CRAFTY eingesetzt, welches eine 'fractional-ply extension' benutzt.

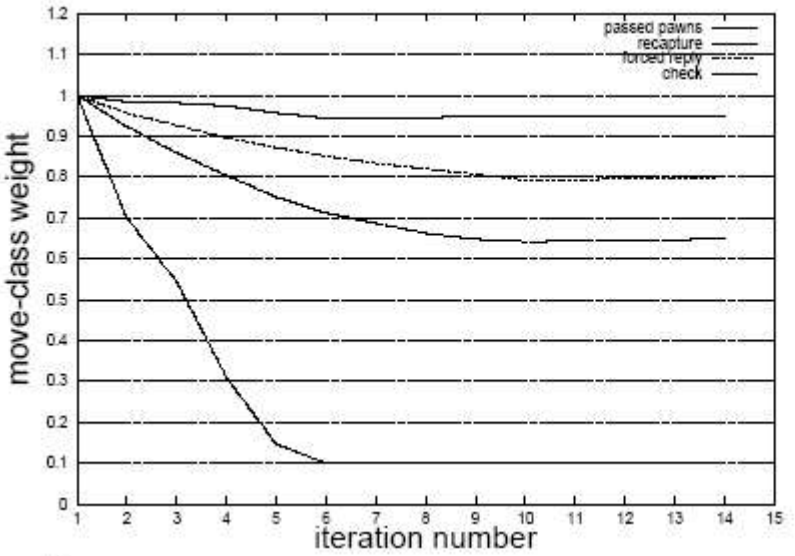
Jeder Zug einer Klasse (4 Zugklassen: Schlagzüge, Schach bieten, Freibauern ziehen, andere) hat eine gewisse Suchtiefe, die ggf. nur ein Bruchteil eines Halbzuges zählt.

Für gesamten Lösungspfad:

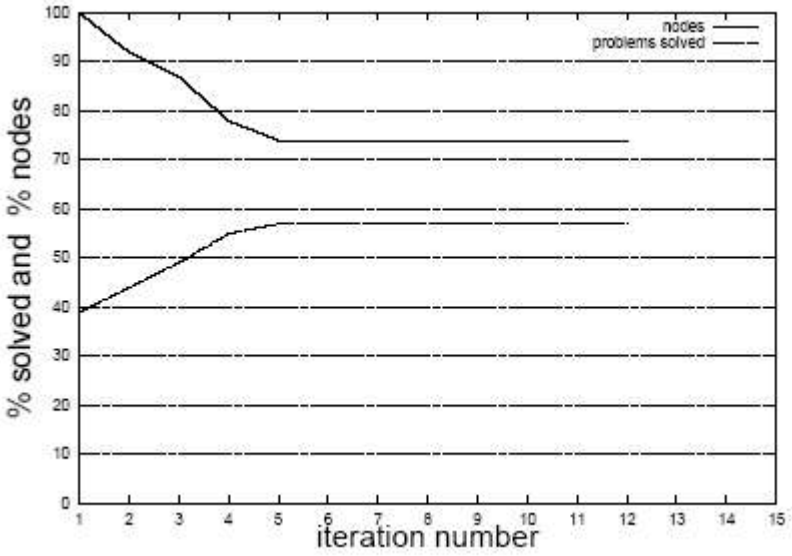
$$D(p, S, \vec{w}) = \sum_{j=1}^{length(S)} w_i \mid i \equiv Class(m_j)$$

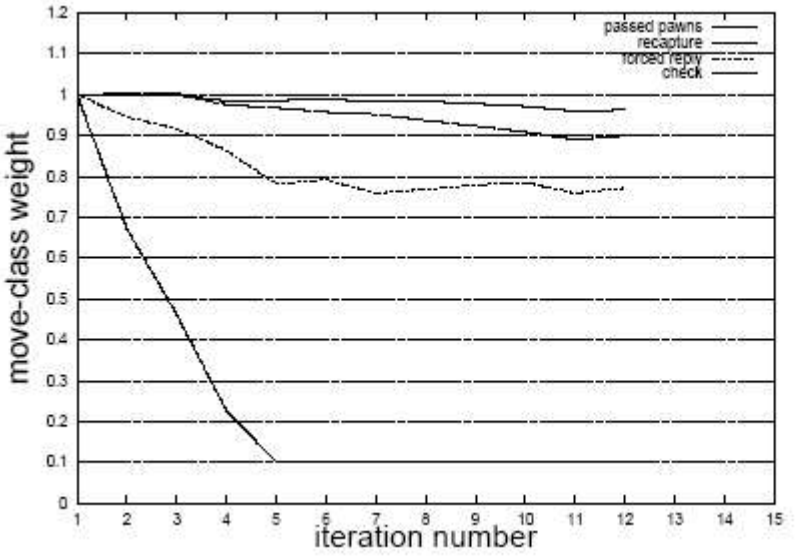
Test auf der ECM-Test Suite





online





offline

'online'-Lernen während des Spiels

Move class	Hand-set	Learned test suite		Learned game play	
		Offline	Online	Middle game	End game
Checks	0.00	0.10	0.10	0.30	0.10
Forced reply	0.25	0.77	0.65	0.47	0.25
Re-captures	0.25	0.90	0.80	0.10	0.10
Passed pawn	0.25	0.96	0.95	0.10	0.15

Match mit verschiedenen Gewichtseinstellungen

vs	C_{games}	C_{050}	C_{hand}	C_{010}	C_{ECM}	C_{100}	Points
C_{games}	-	54.5-45.5	51-49	54-46	59.5-40.5	63.5-36.5	282.5
C_{050}	45.5-54.5	-	53.5-46.5	49.5-50.5	53-47	66-34	267.5
C_{hand}	49-51	46.5-53.5	-	50-50	50.5-49.5	64.5-35.5	260.5
C_{010}	46-54	50.5-49.5	50-50	-	48.5-51.5	64.5-35.5	259.5
C_{ECM}	40.5-59.5	47-53	49.5-50.5	51.5-48.5	-	58-42	246.5
C_{100}	36.5-63.5	34-66	35.5-64.5	35.5-64.5	42-58	-	183.5

Literatur

- K.R.C Greer, P.C. Ojha, D.Bell. A Pattern Oriented Approach to Move Ordering: The Chessmaps Heuristic, ICCAJ, 1999
- Yngvi Björnsson, T.A. Marsland. Learning Extension Parameters in Game-Tree Search, Information Sciences, 2003
- Jonathan Schaeffer. The Games Computers (and People) Play, University of Alberta, 2000