Einführung in die
Künstliche Intelligenz
SS09 - Prof. Dr. J. Fürnkranz

TECHNISCHE UNIVERSITÄT DARMSTADT

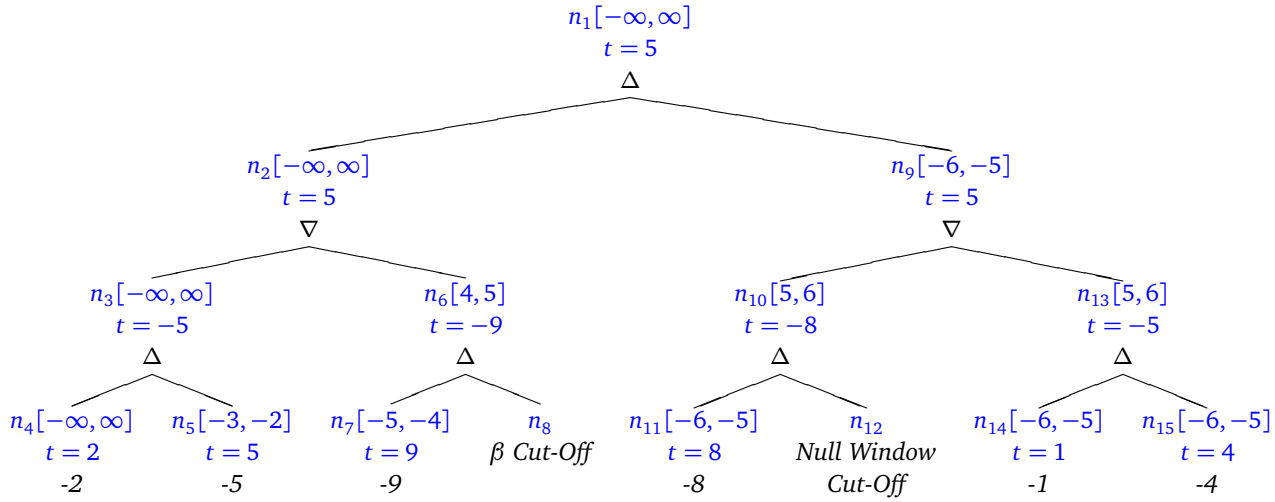
Beispiellösung für das 3. Übungsblatt (02.06.2009)

Aufgabe 3.1 Minimal Window

a) NegaMax und NegaScout erlauben eine simplere Fomulierung des Minimax bzw. Alpha-Beta-Algorithmus (mit Minimal Window), in dem die Konvention benutzt wird, dass die Evaluationswerte immer aus der Sicht des aktiven Spielers dargestellt sind. Somit ist es also nun auch für den MIN-Spieler möglich, durch Maximierung über die Bewertungen der Nachfolgerknoten die optimale Entscheidung zu bestimmen. Für die korrekte „Sichtweise“ wird neben einer Anpassung im ursprünglichen Algorithmus auch die Utility-Funktion verändert, die wie folgt ist:

$$\text{Utility}(n) = \begin{cases} -n, & \text{parent}(n) \text{ is MAX} \\ n, & \text{parent}(n) \text{ is MIN} \end{cases}$$

b) In dem folgenden Baum sind für jeden Knoten die Schranken angegeben, mit denen NegaScout aufgerufen wird.



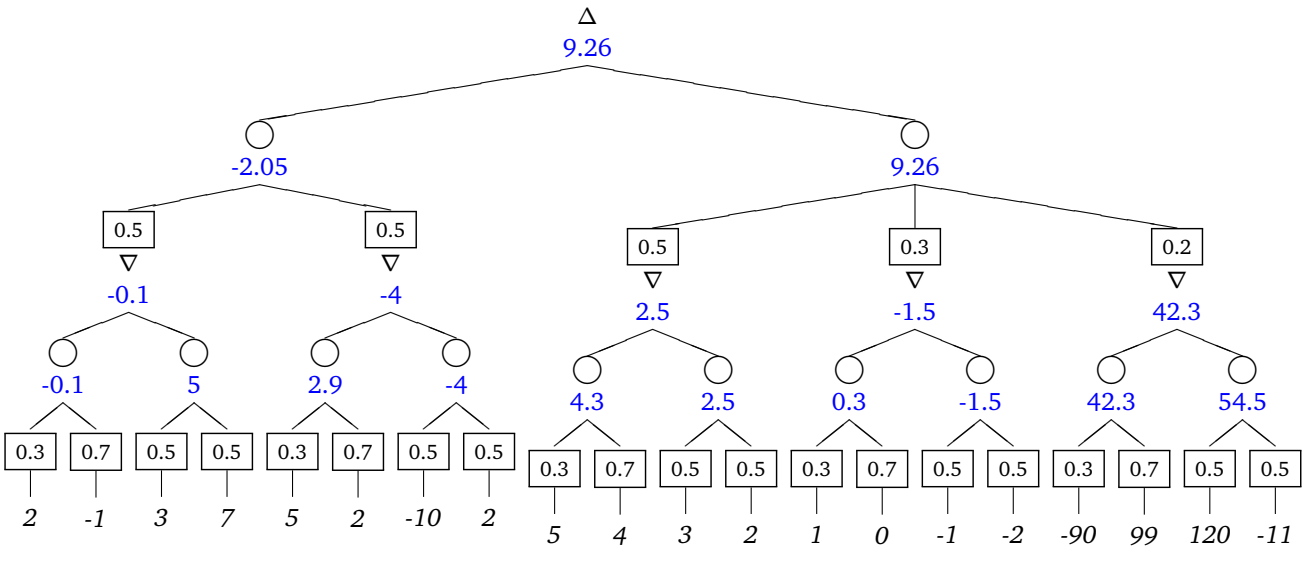
Der Ablauf von NegaScout sieht folgendermassen aus:

NegaScout(n_1,$-\infty$,$+\infty$)
$a=\alpha=-\infty$, $b=\beta=+\infty$
Nachfolger von n_1: $\{n_2, n_9\}$
$t=-$NegaScout(n_2,$-\infty$,$+\infty$)
  $a=\alpha=-\infty$, $b=\beta=+\infty$
  Nachfolger von n_2: $\{n_3, n_6\}$
  $t=-$NegaScout(n_3,$-\infty$,$+\infty$)
    $a=\alpha=-\infty$, $b=\beta=+\infty$
    Nachfolger von n_3: $\{n_4, n_5\}$
    $t=-$NegaScout(n_4,$-\infty$,$+\infty$) $=2$ (Blattknoten)
    $a=2$, $b=3$
    $t=-$NegaScout(n_5,-3,-2) $=5$ (Blattknoten)
    $a=5$, $b=6$

$t = -5$
$a = -5,\ b = -4$
$t = -\text{NegaScout}(n_6,4,5)$
$a = \alpha = 4,\ b = \beta = 5$
Nachfolger von n_6: $\{n_7, n_8\}$
$t = -\text{NegaScout}(n_7,-5,-4) = 9$ (Blattknoten)
$a = 9$
$a \geq \beta$? $9 > 5$ (β Cut-Off, n_8 wird nicht durchsucht)
$t = -9$
$a = -5, b = -4$
$t = 5$
$a = 5,\ b = 6$
$t = -\text{NegaScout}(n_9,-6,-5)$
$a = \alpha = -6,\ b = \beta = -5$
Nachfolger von n_9: $\{n_{10}, n_{13}\}$
$t = -\text{NegaScout}(n_{10},5,6)$
$a = \alpha = 5,\ b = \beta = 6$
Nachfolger von n_{10}: $\{n_{11}, n_{12}\}$
$t = -\text{NegaScout}(n_{11},-6,-5) = 8$ (Blattknoten)
$a = 8$
$a \geq \beta$? $8 > 6$ (Minimal Window Cut-Off, n_{12} wird nicht durchsucht)
$t = -8$
$a = -6,\ b = -5$
$t = -\text{NegaScout}(n_{13},5,6)$
$a = \alpha = 5,\ b = \beta = 6$
Nachfolger von n_{13}: $\{n_{14}, n_{15}\}$
$t = -\text{NegaScout}(n_{14},-6,-5) = 1$ (Blattknoten)
$a = 5,\ b = 6$
$t = -\text{NegaScout}(n_{15},-6,-5) = 4$ (Blattknoten)
$a = 5,\ b = 6$
$t = -5$
$a = -5,\ b = -4$
$t = 5$
$a = 5,\ b = 6$
RETURN(5)

Aufgabe 3.2 Expectiminimax

Der erwartete Minimax Wert ist 9.26.



Aufgabe 3.3 Planen

a)

$at_a(a,p_a)$ % Affe in Position A	$at_s(s_2,p_b)$ % Spielzeug in Position B
$at_b(b,p_b)$ % Banane in Position B	$at_s(s_3,p_c)$ % Spielzeug in Position C
$at_k(k,p_c)$ % Kiste in Position C	hungrig(a) % Der Affe ist hungrig
$at_s(s_1,p_a)$ % Spielzeug in Position A	auf_boden(a) % Der Affe ist auf dem Boden

b)

action: go(A,P)
preconditions: at_a(A,Q), auf_boden(A)
add: at_a(A,P)
delete: at_a(A,Q)

action: push(A,K,P)
preconditions: at_a(A,Q), at_k(K,Q), auf_boden(A)
add: at_a(A,P), at_k(K,P)
delete: at_a(A,Q), at_k(K,Q)

action: throw(A,S,P)
preconditions: at_a(A,Q), at_s(S,Q), auf_boden(A)
add: at_s(S,P)
delete: at_s(S,Q)

action: up(A)
preconditions: at_a(A,P), at_k(K,P), auf_boden(A)
add: auf_kiste(A)
delete: auf_boden(A)

action: down(A)
preconditions: auf_kiste(A)
add: auf_boden(A)
delete: auf_kiste(A)

action: eat(A,B)
preconditions: at_a(A,P), at_b(B,P), auf_kiste(A)
add: satt(A)
delete: hungrig(A), at_b(B,P)

c) Die Formulierung des Ziels lautet: satt(a)

Der kürzeste Plan hierfür ist:
go(a,p_c)
push(a,k,p_b)
up(a)
eat(a)

Faktenmenge: at_a(a,p_a), at_b(b,p_b), at_k(k,p_c), at_s(s_1,p_a), at_s(s_2,p_b), at_s(s_3,p_c), hungrig(a), auf_boden(a)

Aktion: go(a,p_c)

Faktenmenge: at_a(a,p_c), at_b(b,p_b), at_k(k,p_c), at_s(s_1,p_a), at_s(s_2,p_b), at_s(s_3,p_c), hungrig(a), auf_boden(a)

Aktion: push(a,k,p_b)

Faktenmenge: at_a(a,p_b), at_b(b,p_b), at_k(k,p_b), at_s(s_1,p_a), at_s(s_2,p_b), at_s(s_3,p_c), hungrig(a), auf_boden(a)

Aktion: up(a)

neue Fakten: at_a(a,p_b), at_b(b,p_b), at_k(k,p_b), at_s(s_1,p_a), at_s(s_2,p_b), at_s(s_3,p_c), hungrig(a), auf_kiste(a)

Aktion: eat(a)

Faktenmenge: at_a(a,p_b), at_k(k,p_b), at_s(s_1,p_a), at_s(s_2,p_b), at_s(s_3,p_c), satt(a), auf_kiste(a)