18. November 2012

GKI - Hausaufgaben 2

Tao Xu, 343390 - Mitja Richter, 324680 - Björn Kapelle, 320438 - Marcus Weber, 320402

Aufgabe 1

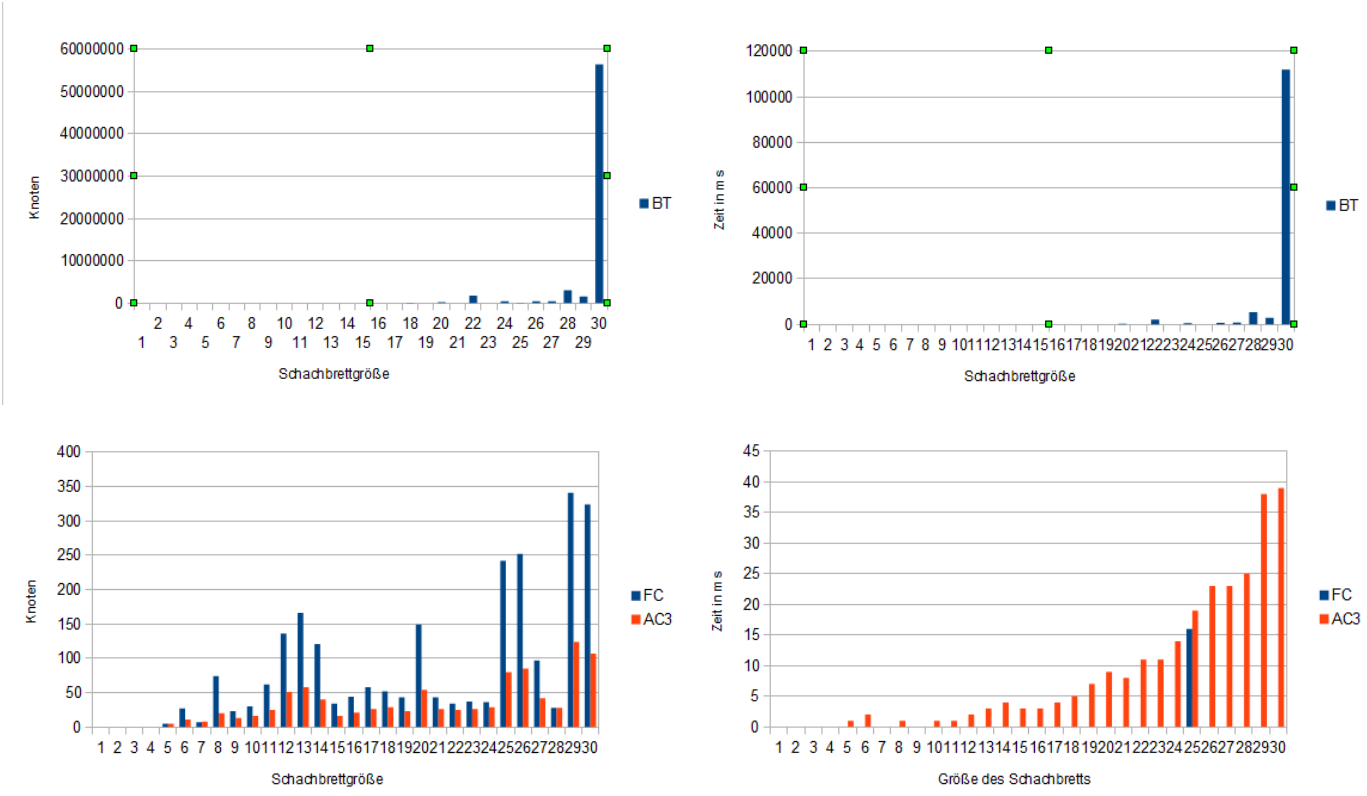
1.a)

javadoc befindet sich unter eight queens/doc/1a.pdf

Backtracking wurde allein in der Klasse QueensBT gelöst. Die Variablen $Q_1 - Q_n$ werden in einem n-stelligem Integer-Array gespeichert. Mithilfe der rekursiven Funktion solveBT, wird dieses der Reihe nach durchiteriert. Sobald für die letzte Stelle des Arrays eine valide Zahl gefunden wird beendet sich der Algorithmus.

Für das Forwardchecking und AC3 benötigen wir eine andere Datenstruktur, da wir die Variablen nicht mehr der Reihe nach durchgehen können. Deswegen wird die Klasse Queen benutzt die die aktuelle Position einer Spalte und deren Domain beinhaltet. Die main-Methode befindet sich in der Klasse QueensFCAC und benötigt als Übergabeparameter einen Modus: „1“ für FC+MRV, „2“ für AC3+MRV und „0“ für normales Backtracking (nur mit eigener Datenstruktur). Des Weiteren ist QueensFCAC ähnlich wie QueensBT aufgebaut, außer der zusätzlichen Funktionen für MRV, FC und AC3 und dass die Variablen nun in einem Array von Queens gespeichert werden, deren Reihenfolge geändert werden kann.

1.b)



Normales Backtracking stößt schnell an seine Grenzen, da ohne Auswahl- und Reduktionsmechanismus sowohl die Bearbeitungszeit, als auch die erzeugten Knoten rasant steigen. Mit Forwardchecking und MRV gibt es eine massive Reduktion in beiden Bereichen, da nur noch ein kleiner Teil des Baumes aufgespannt werden muss. Mit AC3 und MRV wird die Knotenanzahl noch einmal reduziert, d.h. der aufzuspannende Baum ist noch einmal kleiner geworden. Allerdings ist die Berechnung dafür relativ aufwändig, sodass im Vergleich zu FC+MRV die Bearbeitungszeit schon signifikant gestiegen ist. Beide Verfahren haben jedoch massive Vorteile gegenüber Backtracking alleine.

Aufgabe 2

2.a)

Der gegebene Tabelle sieht wie folgt aus.

1		1	2	1	2	1	2
		3	4	3	4	3	4
	2	1	2	1	2	1	2
		3	4	3	4	3	4
1	2	1	2	1	2		
3	4	3	4	3	4	**3**	
1	2	1	2	1	2	1	2
3	4	3	4	3	4	3	4

Aus dieser Tabelle entfernen wir nun Elemente durch die beiden Constraints
$\forall i, j, k \in \{1, \ldots, n\}$ mit $j \neq k : M(i, j) \neq M(i, k)$ (rot gefärbt) und
$\forall i, j, k \in \{1, \ldots, n\}$ mit $j \neq k : M(j, i) \neq M(k, i)$ (blau gefärbt).
Das heißt folgende Tabelle entsteht:

1		~~1~~	2	~~1~~	2	~~1~~	2
		3	4	3	4	~~3~~	4
	2	1	~~2~~	1	~~2~~	1	~~2~~
		3	4	3	4	~~3~~	4
~~1~~	~~2~~	1	2	1	2		
~~3~~	4	~~3~~	4	~~3~~	4	**3**	
~~1~~	~~2~~	1	2	1	2	1	2
3	4	3	4	3	4	~~3~~	4

2.b)

Unsere Heuristik ist: wähle Variable mit den wenigsten konsistenten Werten. Gibt es mehrere, die diese Bedingung erfüllen, so betrachtet man die Variable die die geringste Zeilennummer hat. Sollten auch hier mehrere möglich sein, so betrachtet man diesen Variablen, die mit der kleinsten Spaltennummer. (spätestens diese ist eindeutig)
Wenn wir dies auf unser Problem anwenden wird $M(2, 1)$ (man beachte, dass die Zeilen von unten nach oben und die Spalten von links nach rechts durchnummeriert sind) als erstes gewählt. Wir nehmen diese Heuristik, weil dadurch weniger Zweige erzeugt werden. Zudem wird schneller herausgefunden, ob die Belegung zu einer gültigen Lösung führen kann, als wenn man durch Betrachtung der anderen Variablen, die auf dieses Element durch ein Constraint Einfluss haben, Lösungen ausschließen möchte. Andernfalls müsste man die anderen Variablen betrachten, die auf dieses Element Einfluss nehmen können, und diese durch mehr verschiedene Fälle zu einer Einschränkung führen.

2.c)

Nach unserer Heuristik betrachten wir zunächst $M(2, 1)$. Dort ist lediglich der Eintrag 4 noch möglich. Daraus ergibt sich folgende Tabelle:

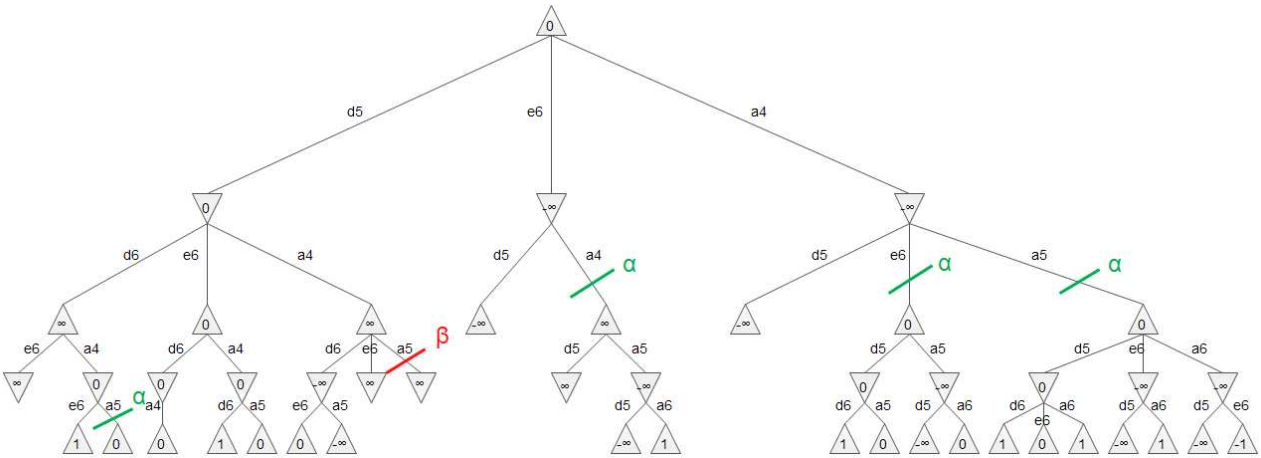
1			2		2		2
		3	4	3	4		4
	2	1		1		1	
		3	4	3	4		4
		1	2	1	2		
	4		~~4~~		~~4~~	**3**	
		1	2	1	2	1	2
3	~~4~~	3	4	3	4		4

Damit ergeben sich folgende neue Wertebereiche:
$M(1, 1) \in \{3\}$, $M(2, 2) \in \{1, 2\}$ und $M(2, 3) \in \{1, 2\}$.

Aufgabe 3

3.a) und b) und d)

Der bewertete Suchbaum mit Cut-Offs sieht folgendermaßen aus:



3.c)

Der Horizonteffekt tritt auf, wenn die Tiefensuche bei einer bestimmten Baumtiefe (hier: $d = 4$) abgebrochen wird. Da der Agent somit unterhalb dieser Tiefe völlig blind ist, kann es sein, dass Blattknoten anders bewertet werden, als wenn man die tieferen Ebenen auch berücksichtigen würde.

Ein Beispiel dafür ist in unserem Fall der Spielverlauf $d5 - e6 - d6 - a4$. Dieses Blatt ist nur mit 0 bewertet, da sowohl gelb als auch rot einen offenen Dreier haben. Wäre die Suchtiefe aber größer, wüsste MAX natürlich, dass er im nächsten Zug gewinnt und würde entsprechend den Knoten mit ∞ bewerten. Damit hätte MAX dann eine sichere Gewinnstrategie.