7. November 2012

GKI - Hausaufgaben 1

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

Aufgabe 1

1.a)

Zustandsraum:
$S = (a_1, a_2, a_3, a_4, a_5, z), a_i \in \{0, 1, 10, 100\}$ und $z \in [0, 127]$
wobei a_i für die jeweils zu bearbeitende Aufgabe steht und $0, 1, 10, 100$ für unbearbeitet(0), Alfons(1), Bernd(10) und Christine(100) (als bearbeitende Personen). Z stehen für die noch aufzuwendende Bearbeitungszeit von Alfons, Bernd und Christine. Es wird also ein 6-stelliges 7bit-Array benötigt.

Startzustand:
$S_0 = (0, 0, 0, 0, 0, 0)$

Zielzustand:
$S_Z = (a_1^Z, a_2^Z, a_3^Z, a_4^Z, a_5^Z, z^Z)$, mit $a_i^Z \neq 0$ und $z^Z = 0$

Aktionen: Die Auswahl des Studenten der die nächste noch nicht bearbeitete Aufgabe bearbeiten soll. Dargestellt mithilfe der Überführungsfunktion *wahl* mit dem Parameter Student (s).
$wahl(s) : S = (a_1, a_2, a_3, a_4, a_5, z) \to S' = (a_1', a_2', a_3', a_4', a_5', z'), s \in \{0, 1, 10, 100\}$
sodass gilt wenn $s \in \{1, 10, 100\}$ (d.h. wenn ein Student für eine Aufgabe ausgewählt wird):

sei a_j das erste $a_i \neq 0$ für mit $<$ geordneten i und / die Ganzzahldivision

- $\forall a_i', i \neq j.\ a_i' = a_i$
- $a_j' = s$ (Belegen der Aufgabe mit Alfons(1), Bernd(10) oder Christine(100))
- $z' = s + z$ (Bilden der neuen Bearbeitungszeit)
- $\frac{z'}{100} \leq 1$ (Christine darf nicht zwei Aufgaben gleichzeitig machen)
- $\frac{(z' \bmod 100)}{10} \leq 2$ (Bernd darf nicht zwei Aufgaben gleichzeitig machen)
- $((z' \bmod 100) \bmod 10) \leq 4$ (Alfons darf nicht zwei Aufgaben gleichzeitig machen)
- $a_1' \neq 100$ (Christine kann Aufgabe 1 nicht)
- $a_3' \neq 100 \wedge a_3' \neq 10$ (Christine und Bernd können Aufgabe 3 nicht)
- $a_4' \neq 100$ (Christine kann Aufgabe 4 nicht)

und wenn $s = 0$ (d.h. wenn kein neuer Student ausgewählt wird und stattdessen eine Zeiteinheit vergeht):

- $\forall a_i'.\ a_i' = a_i$
- $z' = z - sgn(\frac{t}{100}) \cdot 100 - sgn(\frac{(z \bmod 100)}{10}) \cdot 10 - sgn((z \bmod 100) \bmod 10$ (sgn steht hier für Signumfunktion; in diesem Schritt vergeht die Hälfte der durchschnittlichen Bearbeitungszeit)

Pro Zug wird entweder ein Student ausgewählt, der die nächste Aufgabe übernimmt, oder eine Zeiteinheit (in diesem Fall die Hälfte der durchschnittlichen Bearbeitungszeit) vorangeschritten. Damit die zweite Möglichkeit nur in Frage käme, wenn keine validen Belegungen erster Möglichkeit vorhanden sind, kann man die Aktionskosten der zweiten Möglichkeit entsprechend hoch setzen. So könnte man den Algorithmus auch terminieren lassen, indem man zusätzlich zulässige Höchstkosten definiert. Oder indem man z gegen 0 prüft und auf valide Belegungen erster Möglichkeit testet. In der Form ohne diese Zusätze terminiert der Algorithmus erst, wenn er eine valide Belegung für die a_i gefunden hat und $z = 0$.

1.b)

Der Verzweigungsgrad beträgt 4, da wir für die erste Aufgabe 3 Studenten auswählen, oder eine Zeiteinheit verstreichen lassen können.
Ohne andere wie in (1.a) beschriebenen Terminierungsformen ist die maximale Tiefe des Baumes unendlich, da man immer eine Zeiteinheit verstreichen lassen kann.

1.c)

Tiefensuche eignet sich nicht da die maximale Tiefe des Baumes unendlich beträgt. Breitensuche würde dagegen eine (nicht unbedingt optimale) Lösung finden. Best-First-Search (als eigentlich informierte Suche) könnte mithilfe der in 1.b) vorgeschlagenen Aktionskosten schneller zu einer Lösung kommen, die aber nicht unbedingt optimal ist.

1.d)

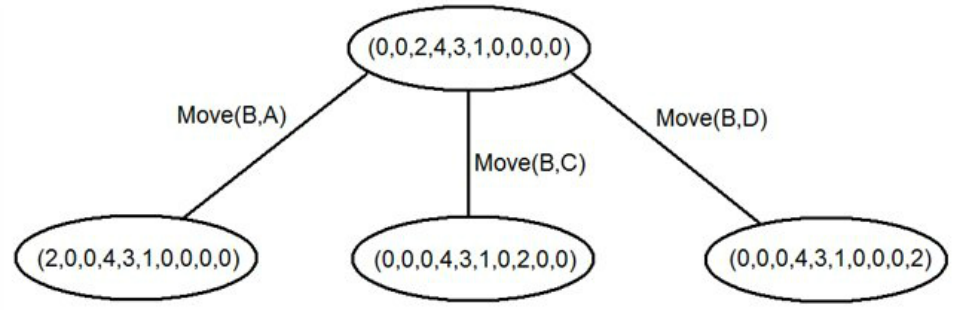
A* für dieses Problem zu verwenden ist problematisch, da es schwierig ist eine Heuristik zu finden die den optimalen Wert nicht zu weit unterschätzt. So kann man zum Beispiel die Aktionskosten für den Fall, dass $s \in \{1, 10, 100\}$ auf die jeweiligen benötigten Zeiteinheiten $1, 2$ und 4 und die Aktionskosten für den Fall dass $s = 0$ auf den höchsten bisherigen Aktionspreis, nämlich 4 setzen. Wenn man jetzt den Wert, den man erhält wenn man errechnet wie viele Zeiteinheiten man mindestens benötigen würde, wenn alle Studenten so schnell wie der schnellste wären (in diesem Fall 2 Zeiteinheiten; 3 Studenten im ersten Durchgang 2 im zweiten), so kommt man hiermit zwar auf eine optimale Lösung, muss aber in ungünstiger Anfangskonstellation einen großen Teil des Baumes durchsuchen.

Aufgabe 2

2.a)

Für die Formulierung eines Suchproblems brauchen wir einen Startzustand, einen Zielzustand, Aktionen und Aktionskosten. Unsere Zustände sind Vektoren der Form $(A_1, A_2, B_1, B_2, B_3, B_4, C_1, C_2, D_1, D_2)$ wodurch gekennzeichnet werden kann welcher Waggon auf welchem Stellplatz ist, $1, 2, 3, 4$ für die 4 Waggons und 0 bedeutet, dass der Stellplatz aktuell leer ist. Der Startzustand ist $(0, 0, 2, 4, 3, 1, 0, 0, 0, 0)$. Der Zielzustand ist $(0, 0, 1, 2, 3, 4, 0, 0, 0, 0)$. Die Aktionen, die wir durchführen können sind abhängig von dem aktuellen Zustand in dem wir uns befinden. Die Aktionen haben aber gemeinsam, dass sie jeweils die Bewegung eines Waggons auf ein anderes Gleisstück beschreiben. Wir unterscheiden die Aktionen Move(X,Y) mit $X, Y \in \{A, B, C, D\}$ und $X \neq Y$. Mit den verschiedenen Kombinationsmöglichkeiten erhalten wir 12 verschiedene Aktionen. Dabei stellen wir allerdings einige Vorbedingungen damit eine Aktion auch durchführbar ist. So muss mindestens ein $X_i \neq 0, i \in \{1, 2\}$, falls X gleich A, B oder C ist, und falls X gleich B soll $i \in \{1, 2, 3, 4\}$ sein und mindestens ein $Y_i = 0$, wobei i wie vorher. Eine weitere Voraussetzung ist, dass die Waggons immer so weit in den Gleisstücken durchrutschen wie möglich, d.h. z.B. ist der Zustand $(0, 1, 0, 2, 3, 4, 0, 0, 0, 0)$ nicht erlaubt. Die Aktionskosten belaufen sich bei allen Aktionen auf 1, da wir möglichst wenig Waggons verschieben wollen und jede Verschiebung gleich aufwendig ist.

2.b)



2.c)

Der Verweigungsgrad ist die maximale Anzahl an Nachfolgerknoten für irgendeinen Knoten in unserem Suchbaum. Die Nachfolgeknoten entsprechen aber den Folgezuständen. Die maximale Anzahl an Folgezuständen ist 12, dies erhalten wir, wenn in jedem Gleisstück ein Waggon liegt. Dann ist nämlich jede Aktion durchführbar. Somit ist der Verzweigungsgrad 12. Die Tiefe des Baumes ist nicht ohne weiteres zu benennen, aber wir kennen mit Gewissheit eine obere Schranke und zwar gilt: Tiefe des Suchbaums $\leq \binom{10}{4} 4! = 5040$. Dies ergibt sich wenn wir alle erdenkbaren Zustände (egal ob für uns zulässig oder nicht) nacheinander erhalten würden. Dabei beschreibt $\binom{10}{4}$ die Anzahl der verschiedenen besetzten Positionen auf allen Gleisstücken und 4! beschreibt die unterschiedliche Positionierung der 4 Waggons auf den Positionen. Da wir Zykel nicht zulassen wollen, erreichen wir auch keine größere Tiefe.

2.d)

Je nachdem auf was man Wert legt, ist entweder die Tiefensuche oder die Breitensuche besser. Der zeitliche Aufwand ist bei der Breitensuche besser, da diese b^d, wobei b dem Verzweigungsgrad entspricht und d der Tiefe der optimalen Lösung entspricht. Dabei ist zu erwähnen, dass die Tiefe der optimalen Lösung die minimale Tiefe des Zielzustands ist. Diese ist mit Sicherheit kein Blatt, da es 3 Vorgängerzustände gibt, die zum Zielzustand führen können und alle drei nicht vorher besucht werden bei einer optimalen Lösung. Dies ist entscheidend, da der zeitliche Aufwand in der Tiefensuche b^m ist, wobei m der maximalen Baumtiefe entspricht. Der Speicheraufwand ist hingegen bei der Tiefensuche besser als bei der Breitensuche, da dieser bei der Breitensuche b^d entspricht und bei Tiefensuche lediglich $b \cdot\ m$. Der exponentielle Anstieg bei der Breitensuche hat einen deutlich stärkeren Effekt als der lineare Anstieg bei der Tiefensuche,da d mindestens 8 ist (weil jeder Waggons mindestens 2 mal bewegt werden muss, sonst kann der Zielzustand nicht erreicht werden). b ist gleich 12 und m ist maximal 5040. Es gilt, aber dass $12^8 > 12 \cdot\ 5040$. Somit hat man auch einen Zahlenbeleg für unsere Behauptung. Daher liegt es in den Augen des Betrachters welches Verfahren besser geeignet, je nach Interessenlage.

Aufgabe 3

3.a)

Die Iterative Tiefensuche startet bei einem Wurzelknoten und führt dort im ersten Schritt eine limitierte Tiefensuche mit minimaler Suchtiefe. In jedem Iterationsschritt wird nur erneut eine limitierte Tiefensuche durchgeführt, während die Suchtiefe um 1 erhöht wird. Abgebrochen wird, sobald eine konsistente Belegung gefunden wird, oder die maximale Suchtiefe erreicht ist. Die Performance des Algorithmus ist ähnlich der normalen Tiefensuche, allerdings ist er durch die Iteration die Optimalität betreffend auch ähnlich gut wie die Breitensuche. Da jede Iteration teilweise gleiche Knoten aufgespannt werden müssen wirkt sich das (geringfügig) negativ auf die Laufzeit aus. Kombiniert man die Iterative Tiefensuche mit dem Minmax-Algorithmus, kommt der positive Effekt hinzu, dass man zu jedem Zeitpunkt eine Lösung abrufen kann.

3.b)

Die Formel für alle generierten Knoten der Iterativen Tiefensuche ist:
$N_{ITS} = \sum_{i=0}^{d} (d + 1 - i)b^i$, mit d = Tiefe und b=Verzweigung
Einsetzen ergibt:
$N_{ITS} = \sum_{i=0}^{5^5} (5^5 + 1 - i)35^i \approx 1,08 \cdot 10^{193}$

Für die limitierte Tiefensuche gilt:
$N_{LTS} = \sum_{i=0}^{d} b^i$
Einsetzen ergibt:
$N_{LTS} = \sum_{i=0}^{5^5} 35^i \approx 1,05 \cdot 10^{193}$

Das ergibt für den Overhead $\approx 2,8\%$

3.c)

Je höher der Verzweigungsfaktor ist, desto kleiner ist der Overhead. Also eignet sich iterative Tiefensuche eher bei hohem Verzweigungsfaktor. Dieser Effekt kommt daher, dass bei Bäumen mit hohem Verzweigungsfaktor die Knoten in der maximale Tiefe (also den noch nicht erzeugten Knoten), einen sehr großen Teil des Baumes ausmachen. Je mehr Iteration man macht, desto größer wird dieser Anteil.

Aufgabe 4

4.a)

Schritt	Expandierter Knoten	Queue	Anmerkungen
1	A(35)	AC(30), AB(40)	keine
2	AC(30)	ACZ(35), AB(40), ACD(40)	ACA entfernt (Zykel) ACB(65) entfernt (DP)
3	ACZ(35)	-	Algorithmus terminiert

4.b)

A^* liefert nur dann mit Sicherheit die optimale Lösung, wenn die zugrundeliegende Heuristik zulässig ist. In unserem Fall ist die Heuristik nicht zulässig, denn sie überschätzt die Kosten für A ($35 > 30$) und für B ($35 > 25$). Und in der Tat erhalten wir nicht das optimale Ergebnis.

4.c)

Wir ändern die Heuristik für A und für B, so dass $h(A) = 25$ und $h(B) = 20$ gilt. Dann folgt:

Schritt	Expandierter Knoten	Queue	Anmerkungen
1	A(25)	AB(25), AC(30)	keine
2	AB(25)	ABC(25), ABD(30)	ABA entfernt (Zykel) AC(30) entfernt (DP)
3	ABC(25)	ABCZ(30), ABD(30)	ABCA entfernt (Zykel) ABCB entfernt (Zykel) ABCD(35) entfernt (DP)
4	ABCZ(30)	-	Algorithmus terminiert

Die Heuristik ist jetzt zulässig und A^* liefert die optimale Lösung.

Aufgabe 5

Bei den Aufgaben a) bis c) ist die Zulässigkeit verschiedener Metriken als Heuristik zu zeigen. Eine Heuristik h ist zulässig, wenn $0 = h(X) = h^*(X)$ für alle Knoten X, wobei h^* die tatsächlichen Kosten bezeichnet.

5.a)

Wir betrachten als heuristische Funktion h die euklidische Metrik. Die euklidische Metrik ist definiert als $d_{eukl}(x, y) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, wobei $x := (x_1, y_1)$ und $y := (x_2, y_2)$.
Da $(x_2 - x_1)^2 + (y_2 - y_1)^2 \geq 0$, $\forall x_1, x_2, y_1, y_2 \in \mathbb{R}$, ist die Wurzel dieses Terms definiert und da diese wiederum ein nichtnegatives Ergebnis ausgibt gilt: $0 \leq h(X)$ für alle möglichen zweidimensionalen Koordinaten (Längen- und Breitengrade). Zudem beschreibt die euklidische Metrik die Luftlinienentfernung zwischen zwei Punkten, somit gilt auch: $h(X) \leq h^*(X) \Rightarrow$ die euklidische Metrik ist zulässig.

5.b)

Wir betrachten als heuristische Funktion h die Maximum-Metrik.
Da $|x_2 - x_1| \geq 0$ und $|y_2 - y_1| \geq 0$, $\forall x_1, x_2, y_1, y_2 \in \mathbb{R}$, ist $\max(|x_2 - x_1|, |y_2 - y_1|)$ ebenfalls nichtnegativ. Somit gilt: $0 \leq h(X)$ für alle möglichen zweidimensionalen Koordinaten (Längen- und Breitengrade).
Zudem gilt: $d_{max}(x, y) \leq d_{eukl}(x, y)$ (siehe d.) und somit gilt auch $h(X) \leq h^*(X) \Rightarrow$ die Maximum-Metrik ist zulässig.

5.c)

Wir betrachten als heuristische Funktion h die Manhattan-Metrik.
Diese Metrik ist nicht zulässig. Dies zeigen wir durch ein Beispiel an dem man erkennen kann, dass die Manhattan-Metrik für unser Problem die tatsächlichen Kosten überschätzen kann. Dazu seien zwei Städte an den Koordinaten (0,0) und (1,1) die auf direktem Weg (Luftlinie) miteinander durch eine Straße verbunden sind. Die Entfernung zwischen diesen Städten ist demnach $\sqrt{2}$, aber die Manhattan-Metrik gibt $|1 - 0| + |1 - 0| = 2$ als Ergebnis aus. Somit ist die Manhattan-Metrik für unser Beispiel nicht zulässig.

5.d)

Für gegebene reelle Wertepaare $x := (x_1, y_1)$ und $y := (x_2, y_2)$ gilt: $d_{max}(x, y) \overset{1.}{\leq} d_{eukl}(x, y) \overset{2.}{\leq} d_{man}(x, y)$

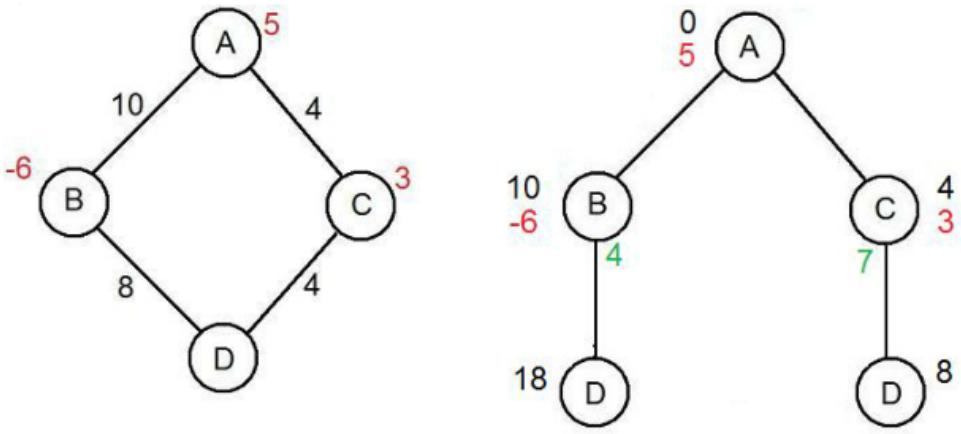
Beweis 1.: Ohne Beschränkung der Allgemeinheit sei: $d_{max}(x, y) = \max(|x_2 - x_1|, |y_2 - y_1|) = |x_2 - x_1|$. (Der andere Fall würde genau analog verlaufen)
$|x_2 - x_1| = \sqrt{(|x_2 - x_1|)^2} = \sqrt{(x_2 - x_1)^2} \leq \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
Der vorletzte Schritt ergibt sich aus der Tatsache, dass $(x_2 - x_1)^2 \geq 0$ und somit der Betrag hinfällig ist. Der letzte Schritt ergibt sich aus der Monotonie der Wurzelfunktion und der Tatsache, dass $(x_2 - x_1)^2 + (y_2 - y_1)^2 \geq (x_2 - x_1)^2$.

Beweis 2.: Seien $a, b \in \mathbb{R}_0^+$. Dann gilt: $a^2 + b^2 = a^2 + 2ab + b^2 = (a+b)^2$ und somit $\sqrt{a^2+b^2} \leq \sqrt{(a+b)^2} \overset{a,b\geq 0}{=} (a+b)$ (Es wurde wieder die Monotonie der Wurzelfunktion benutzt).
Sei nun $a := |x_2 - x_1| \geq 0$ und $b := |y_2 - y_1| \geq 0$ und damit folgt die Behauptung.

5.e)

Die Manhattan-Metrik ist nicht die geeignetste, da sie nicht zulässig ist und da die euklidische Metrik die Maximum-Metrik dominiert, ist die euklidische Metrik für unser Problem die geeignetste, da ihre Werte näher an den tatsächlichen Kosten liegen.

5.f)



An diesem Beispiel kann man sehen, dass wenn man den Algorithmus durchführen möchte, um einen kürzesten Weg von D nach A zu finden, man alle beiden Wege überprüfen wird. Allerdings würde man dies nicht tun, wenn statt -6 als heuristischer Wert 0 eingesetzt werden würde, da der Gesamtwert nicht mehr 4 sondern 10 wäre und somit in Anbetracht der Gesamtkosten des Pfades A-C-D gar nicht geprüft werden müsste. Bei größeren Beispielen kann dies zu erheblichen Teilpfaden führen, die man weglassen könnte und somit effektiver arbeiten.

5.g)

Die Vollständigkeit des Algorithmus ist weiterhin gegeben, da wenn ein Pfad nicht zu einem Zielknoten fihrt, ein anderer geprüft werden würde und dies so lange bis mindestens ein Punkt erreicht wird an dem Start- und Zielknoten wie gewünscht sind. Sollte kein derartiger Pfad existieren, kann der Algorithmus keinen Pfad finden und von daher kann man solche Fälle vernachlässigen.
Die Optimalität wird ebenfalls nicht beeinträchtigt, da die heuristische Werte nur Schätzwerte sind, die ein schnelleres Finden des besten Weges ermöglichen sollen. Wenn wir negative heuristische Werte zulassen, finden wir den gewünschten Weg evtl. nicht schneller, aber solange wir Heuristiken benutzen, die die tatsächlichen Kosten unterschätzen, erreichen wir trotzdem die beste Lösung.
Dies gilt, weil für die Gesamtkosten der Wege am Ende die heuristischen Werte keinen Einfluss haben. Die Überprüfung anderer - vielleicht besserer - Wege wird gewährleistet durch die Unterschätzung, denn somit wird kein potenzieller Weg durch die heuristischen Werte als „nicht weiter zu überprüfen“ gekennzeichnet.

Aufgabe 6

6.a)

w=weiß, g=grau, s=schwarz

Schritt	A	B	C	D	E
Startbelegung Startkonflikte(10)	w w(10)g(7)s(7)	w w(10)**g(6)**,s(6)	w w(10)g(7)s(7)	w w(10)g(7)s(7)	w w(10)g(7)s(7)
1 (6)	w w(6)g(5)**s(4)**	g w(10)g(6),s(6)	w w(6)g(5)s(4)	w w(6)g(5)s(4)	w w(6)g(5)s(4)
2 (4)	s w(6)g(5)s(4)	g w(7)g(4)s(5)	w w(4)g(3)**s(2)**	w w(4)g(4)s(4)	w w(4)g(3)s(2)
3 (2)	s w(4)g(3)s(2)	g w(4)g(2)s(3)	s w(4)g(3)s(2)	w w(2)g(2)s(2)	w w(2)g(2)s(2)
4 (1)	s w(2)g(2)s(1)	g w(3)g(1)s(3)	s w(2)g(2)s(1)	w w(1)g(1)s(1)	w w(1)g(1)s(1)

Schritt	F	G
Startbelegung Startkonflikte(10)	w w(10)g(8)s(8)	w w(10)g(8)s(8)
1 (6)	w w(6)g(4)s(4)	w w(6)g(4)s(4)
2 (4)	w w(4)g(3)s(4)	w w(4)g(2)s(2)
3 (2)	w w(2)**g(1)**s(2)	w w(2)g(1)s(2)
4 (1)	g w(2)g(1)s(2)	w w(1)g(2)s(2)

Das Ergebnis hat immer noch einen Konflikt, d.h. der Algorithmus ist zu keiner Lösung gekommen.

6.b)

Die Tie-Break-Regel muss dahingehend geändert werden, dass sie den den höchsten lexikographischen Folgezustand bevorzugt.

In 6.a) ist das Problem, dass sehr früh A und C belegt werden. Dann ist aber D durch A und B festgelegt und E ist durch B und C festgelegt. Die Constraintverletzung zwischen D und E kann dann nicht mehr in einem Schritt gelöst werden und die lokale Suche bleibt somit in einem lokalen Minimum stecken.
Die geänderte Tie-Break-Regel belegt nun hingegen A und C später und dafür E früher. Somit kommt es zu keinem Konflikt zwischen D und E. Die spätere Belegung von A und C führt zu keinen Problemen, denn deren Konflikte mit F bzw. G können immer in einem Schritt gelöst werden, da F und G nur zwei Constraints haben.