Multiple-Choice Questions: Correct answers give positive points, but wrong answers give negative points!)

Teil (Part) 1 **40 Punkte (points)**

a) Modellieren Sie ein neuronales Netz mit maximal einem inneren Layern, welches zwei binäre Eingangssignale I_1, I_2 sowie drei Ausgangssignale O_1, O_2, O_3 besitzt und dessen Signale sich wie in der folgenden Wahrheitstabelle verhalten.

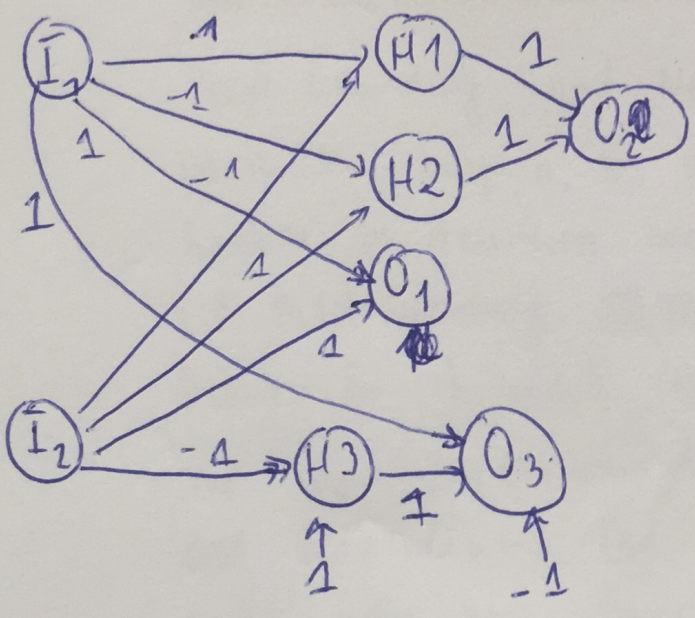
(Construct a neural network with at most one inner layer, which has two binary input signals I_1, I_2 and three output signals O_1, O_2, O_3. The signals should behave according to the entries of the following truth table.)

OR XOR ¬IF ¬(¬A∨B) A∧¬B

I_1	I_2	O_1	O_2	O_3
0	0	0	0	0
0	1	1	1	0
1	0	1	1	1
1	1	1	0	0

Verwenden Sie für jedes Neuron folgende Aktivierungsfunktion: (For each neuron, use the following activation function:)

$$g(x) = \begin{cases} 1 & \text{wenn (if) } x \geq 1\,, \\ 0 & \text{sonst (otherwise).} \end{cases}$$



6 Punkte (points)

b = branching Faktor d = Tiefe der "seichtesten" Lösung
m = Maximale Tiefe des Suchraums

b) (Beschreiben Sie die Eigenschaften der *Depth-first iterative deepening search (DFIDS)* und der *Depth-first search (DFS)* in Big-O Notation. Erklären Sie die Unterschiede der Suchvarianten.)

Describe the properties of *Depth-first iterative deepening search (DFIDS)* and *Depth-first search (DFS)* and provide the worst case complexities in Big-O notation. Explain the differences between the two search procedures.

6 Punkte (points) ✓

DFIDS: Räumliche Komplexität $O(bd)$
zeitliche Komplexität $O(b^d)$
Optimal, wenn Schrittkosten Konstant. Vollständig, wenn Branching Faktor nicht unendlich

DFS: R.K. $O(bm)$, Z.K. $O(b^m)$
Nicht optimal und nicht vollständig

c) Nennen Sie die 4 Komponenten der *Task Description* eines Agents. Geben Sie ein Beispiel für jede Komponente.

(Name the 4 components of the *task description* of an agent. Give an example for each component.)

4 Punkte (points) ✓

Taxi

Performance Measure — Komfort
Environment — Straße
Actuators — Lenkrad
Sensors — Kameras

d) Was ist *Deep Learning* und was wird benötigt um es anwenden zu können? Geben Sie Beispiele wo *Deep Learning* gut funktioniert, und erklären Sie welche Schwächen es hat.

(What is *Deep Learning* and what is needed in order to apply them? Give examples for which *Deep Learning* works well, and explain its shortcomings.)

5 Punkte (points) ✓

Deep Learning sind Neuronale Netzwerke mit vielen versteckten Layern. Es werden viele Daten benötigt, um das System zu trainieren. Deep Learning funktioniert gut bei z.B Bilderkennung, d.h. bei nicht strukturierten Daten. Es ist nicht immer einfach, das System zu verfeinern, und die Anzahl der Layer und Gewichte ist relativ komplex. Die Entscheidungen von DL sind teilweise auch nicht oder nur schwer nachvollziehbar

e) Welche Funktionen können (in Abhängigkeit von der Anzahl der versteckten Ebenen i.e. hidden layers) in Single- und Multi-Layer Perzeptronnetzwerken dargestellt werden?

(What functions can be represented in single- and multi-layer perceptron networks (depending on the number of hidden layers i.e. hidden layers)?)

4 Punkte (points)

Single Layer Perceptrons: Linear separierbare Funktionen ✓

2 Layer: Kontinuierliche Funktionen

3 Layer: Alle Funktionen.

f) Beweisen oder widerlegen Sie, dass wenn beliebige Heuristiken h_1 und h_2 monoton (konsistent) sind, dann ist

(Prove or refute that, if any given heuristics h_1, h_2 are both monotone (consistent), then)

$$h(n) = \max\{h_1(n), h_2(n)\}$$

auch monoton (Konsistent). $h(n) \leq h(n') + c(n, a, n')$

(is also monotone (consistent).)

6 Punkte (points) ✓

$h(n) = \max\{h_1(n), h_2(n)\}$

$\leq \max\{h_1(n) + c(n, a, n'), h_2(n') + c(n, a, n')\}$ follows from h_1, h_2 monotone properly in IR

$= \max\{h_1(n'), h_2(n')\} + c(n, a, n')$

$= h_{\max}(n') + c(n, a, n')$ def. $h(n')$

Die Heuristik $h(n)$ ist monoton (konsistent).

g) Beschreiben Sie das Funktionsprinzip von *Hill Climbing*. Welche Lösungen werden erzeugt und unter welchen Bedingungen findet man mit Hill Climbing keine Lösung?

(Describe the working principle of *Hill Climbing*. What solution are generated and under which conditions does *Hill Climbing* find no solution?)

4 Punkte (points) 3

Bei Hill Climbing wird eine ~~Lösung~~ Zustand zufällig erzeugt. In einer Schleife wird von dem Zustand solange immer die beste Lösung aus der Nachbarschaft gesucht, bis keine Verbesserung mehr möglich ist. Es kann passieren, dass ~~es~~ sich bei der Lösung um ein lokales Maximum handelt, dies keine gültige Lösung ist.

39

Teil (Part) 2

40 Punkte (points)

a) Wir betrachten das Dreifärbbarkeitsproblem für folgende Karte:
(We consider the three-coloring problem for the following map:)

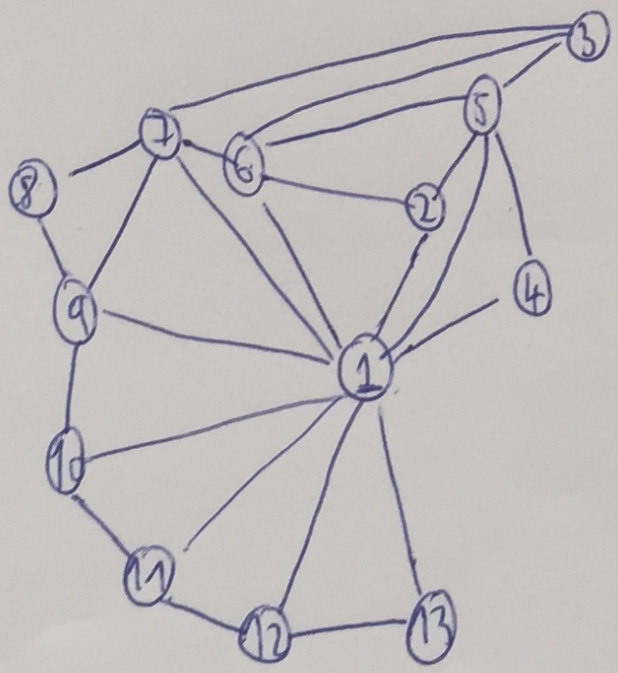
Gegeben ist folgende Beschreibung als *Constraint Satisfaction Problem* (CSP):
(The *Constraint Satisfaction Problem* (CSP) is described as follows:)

Variablen (Variables):	$\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13\}$
Domänen (Domains):	$D_i = \{\text{red}, \text{green}, \text{blue}\}$, $1 \leq i \leq 13$
Constraints:	Benachbarte Regionen sollen unterschiedliche Farben haben. Zur Klarstellung : 3 und 7 grenzen aneinander. (Adjacent states shall have different colors. For clarification: 3 and 7 are adjacent.)

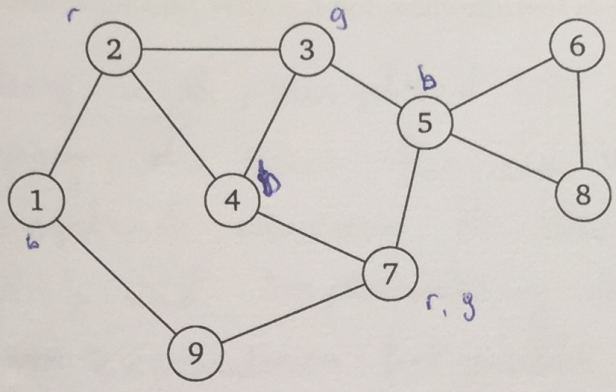
Konstruieren Sie den zugehörigen *Constraint-Graph*.
(Construct the corresponding *constraint graph*.)

6

6 Punkte (points)



b) Wir betrachten das Dreifärbbarkeitsproblem für folgenden Graphen:
(Consider the three-colorability problem for the following graph:)



(a) Ausgehend von der partiellen Zuordnung "3 = grün, 4 = blau", welchen Wert würde der Variablen 5 durch die *Least-Constraining-Value Heuristik* zugeordnet werden?
(Based on the partial assignment "3 = green, 4 = blue", which value would be assigned to the variable 5 by the *least-constraining-value heuristic*?)

blau

(b) Betrachten Sie die partielle Zuordnung "1 = blau, 2 = rot, 3 = grün". Welche Variable würde durch die *Minimum-Remaining-Values Heuristik* als nächstes ausgewählt werden?
(Consider the partial assignment "1 = blue, 2 = red, 3 = green". Which variable would be assigned next using the *minimum-remaining-values heuristic*?)

4

(c) Welche Variable würde durch die *Degree Heuristik* als erstes einen Wert aus ihrer Domain zugeordnet bekommen?
(Which variable would be assigned a value from its domain first by the *degree heuristic*?)

5 (degree 4)

6 Punkte (points) 6

c) Erkären Sie *Backtracking Search* für CSP. Warum ist sie im Allgemeinen nicht besonders effizient?

(Explain *Backtracking Search* for CSP. Why is it not really efficient in general?)

4 Punkte (points) 3

Bei Backtracking wird, ausgehend von einem Zustand ohne Variablenbelegung, der Wert für eine Variable festgelegt und davon ausgehend rekursiv der Wert einer weiteren Variable. Je nach Heuristik und Implementierung kann ein Fehler erst beim ~~Expandieren~~ Expandieren festgestellt werden → Backtracking. Im schlimmsten Fall hat Backtracking eine ~~exponentielle~~ exponentielle Laufzeit, findet am Ende aber eine Lösung, wenn es eine gibt.

→ alle (d^n) Belegungen werden überprüft (gesamter Baum aufgespannt)

d) Welche drei Probleme gibt es beim Planen bezüglich dem Schlussfolgern über Handlungen? Erklären Sie diese kurz.

(What are the three problems that occur in reasoning about actions? Give a short explanation for each of them.)

6 Punkte (points) 6

Frame Problem: Wie werden Aktionen modelliert, die keinen Einfluss auf Dinge haben (z.B. Axiome)

Qualifikation Problem: Welche Vorbedingungen müssen erfüllt sein, sodass eine Aktion erfolgreich ist.
z.B. Flugzeug fliegt von A nach B. Frau muss in A sein, A muss einen Flugplatz haben, B muss eine Stadt sein, ... → konkrete Konzeptrealisierung der Dinge

Ramifikation Problem: Wie werden implizite Effekte modelliert?
Auto fährt von A nach B, so auch der Motor, die Hitze, die Insassen ...

e) Formalisieren Sie in STRIPS Syntax eine Aktion *Install* um ein Programm von einem Server herunterzuladen um es auf einem anderen Server zu installieren.

(Formalize in STRIPS syntax the action *Install* for downloading a program from some server and installing it on another server.)

- Vorbedingung für diese Aktion ist, dass das Program angefordert wird, dass das Program am Quellserver vorhanden ist und dass dem Quellserver vertraut wird.
  (The precondition for this action is that the program is requested, that the program is available on the source server, and that the source server is trusted.)
- Der Effekt soll sein, dass das Programm nicht mehr angefordert wird und dass das Programm jetzt auch am Zielserver vorhanden ist.
  (The effect is that the program is no longer requested and that the program is now also available on the target server.)
- "Programm" soll erster Parameter, "Quellserver" und "Zielserver" sollen zweiter bzw. dritter Parameter sein.
  ("Program" is the first parameter, "source server" and "target server" are the second and third parameter, respectively.)
- Die Parameter sollen in der Vorbedingung entsprechend ihrem Typus geprüft werden.
  (The parameters need to be checked according to their type.)

Verwenden Sie für die Modellierung die einstelligen Prädikate *Program*, *Server*, *Requested* und *Trusted*, sowie das zweistellige Prädikat *Source*.

(For modeling the action use the unary predicates *Program*, *Server*, *Requested*, and *Trusted*, as well as the binary predicate *Source*.)

Die Bedeutung der Prädikate ist wie folgt:

(The meaning of the predicates is as follows:)

- $Program(x)$: x ist ein Programm, ($Program(x)$: x is a program,)
- $Server(x)$: x ist ein Server, ($Server(x)$: x is a server,)
- $Trusted(x)$: x wird vertraut, ($Trusted(x)$: x is trusted,)
- $Requested(x)$: x wird angefordert, und ($Requested(x)$: x is requested, and)
- $Available(x, y)$: x ist auf y vorhanden. ($Available(x, y)$: x is available on y.)

6 Punkte (points) 6

Action (Install (x, y, z),
PRECOND: Server (y) ∧ Server (z) ∧ Trusted (y) ∧ Requested(x) ∧
Available (x, y) ∧ Program (x),
EFFECT: ¬Requested (x) ∧ Available (x, z))

f) Erklären Sie, welche Knotenarten es in *Decision Networks* gibt und wozu diese verwendet werden.

(Describe the node types used in *decision networks* and explain what they are used for.)

6 Punkte (points)

6

Chance Nodes: Repräsentieren Zufallsvariablen; können ihren Wert abhängig einer stochastischen Verteilung haben. Oval. ○

Decision Nodes: Repräsentieren Punkte, an dem der Entscheidungsträger eine Entscheidung trifft. Diese kann die Chance Nodes beeinflussen. □ Rechteckig

Utility Nodes: Repräsentieren eine Utility Function des Agenten. Diese Knoten haben als Eltern alle Variablen, die die Utility beeinflussen. ◇ Diamantförmig

g) Kreuzen Sie Zutreffendes an:
(Check the correct answers:)