

Heuristiken

Carsten Gips (HSBI)

Unless otherwise noted, this work is licensed under CC BY-SA 4.0.

VARIABLES: Variablen-Sortierung, Welche Variable soll betrachtet werden?

```
def BT_Search(assignment, csp):  
    if complete(assignment): return assignment  
    var = VARIABLES(csp, assignment)  
    for value in VALUES(csp, var):  
        if consistent(value, var, assignment, csp):  
            assignment += {var = value}  
  
            if INFERENCE(csp, assignment, var) != failure:  
                result = BT_Search(assignment, csp)  
                if result != failure: return result  
  
            assignment -= {var = value}  
  
    return failure
```

Quelle: Eigener Code basierend auf einer Idee nach (Russell und Norvig 2020, S. 176, Fig. 5.5)

Minimum Remaining Values (MRV): (vgl. (Russell und Norvig 2020, 177))

- Wähle Variable mit wenigsten freien Werten

VARIABLES: Gleichstand bei MRV

```
def BT_Search(assignment, csp):  
    if complete(assignment): return assignment  
    var = VARIABLES(csp, assignment)  
    for value in VALUES(csp, var):  
        if consistent(value, var, assignment, csp):  
            assignment += {var = value}  
  
            if INFERENCE(csp, assignment, var) != failure:  
                result = BT_Search(assignment, csp)  
                if result != failure: return result  
  
            assignment -= {var = value}  
  
    return failure
```

Quelle: Eigener Code basierend auf einer Idee nach (Russell und Norvig 2020, S. 176, Fig. 5.5)

Gradheuristik: Erweiterung von *MRV* bei *Gleichstand* (vgl. (Russell und Norvig 2020, 177))

- Wähle Variable mit meisten Constraints auf offene Variablen

VALUES: Werte-Sortierung, Welchen Wert soll ich ausprobieren?

```
def BT_Search(assignment, csp):  
    if complete(assignment): return assignment  
  
    var = VARIABLES(csp, assignment)  
    for value in VALUES(csp, var):  
        if consistent(value, var, assignment, csp):  
            assignment += {var = value}  
  
            if INFERENCE(csp, assignment, var) != failure:  
                result = BT_Search(assignment, csp)  
                if result != failure: return result  
  
            assignment -= {var = value}  
  
    return failure
```

Quelle: Eigener Code basierend auf einer Idee nach (Russell und Norvig 2020, S. 176, Fig. 5.5)

Least Constraining Value (LCV): (vgl. (Russell und Norvig 2020, 177))

- Wähle Wert, der für verbleibende Variablen die wenigsten Werte ungültig macht

- Verbesserung der BT-Suche mit Heuristiken: MRV, Gradheuristik, LCV



Unless otherwise noted, this work is licensed under CC BY-SA 4.0.