GWV Hausaufgaben 2

Benjamin Cordt
Paul Hölzen

28. Oktober 2017

Aufgabe 2.4

- *state* $:< P, F >$ wobei P eine Matrix ist, deren Koordinaten Orte in der Wohnung repräsentieren. Die Werte sind dabei Element der Menge $\tilde{P} = F \cup x \cup 0$. F ist die Menge von Möbeln, die Werte x und 0 stehen für Plätze auf denen keine Möbel stehen können und freie Stellen.
  Die Menge F besteht aus Zwei-Tupeln mit einem eindeutigen Bezeichner b an der ersten und zwei Zahlen $x, y \in \mathbb{N}$ an der zweiten und dritten Stelle, die die benötigte Fläche beschreiben.

  Der Startzustand besteht dann aus P, einer Matrix, die eine leere Wohnung darstellt, und F der Menge aller Möbel, die darin aufgestellt werden sollen. Bei einem Zustandsübergang wird ein Element f aus F entfernt und benachbarte Werte 0 der Matrix P werden entsprechend der Maße x und y, auf den Bezeichner b gesetzt.
  Gültige Zustände seien nur solche, bei denen Stühle benachbart zu Tischen sind und Platz vor der Tür bleibt etc. Das Ziel ist erreicht, wenn die Menge F leer ist. Das bedeutet es gibt vielleicht keinen Zielzustand, wenn z.B. zu viele Möbel für eine kleine Wohnung eingeplant wurden oder alles voller Türen ist.
  Der am besten geeignete Suchalgorithmus ist die Breitensuche, da viele Teilbäume kein Ziel enthalten werden und die Tiefensuche hier potentiell viel Zeit verschwendet.

- *state* $:< (c_1, w_1), ..., (c_n, w_n) >$ wobei es eine Menge an Komponenten des Hauses gibt, deren Elemente als Tupel mit Elementen einer Menge

an Arbeitern kombiniert werden können.
Seien z.B. die Komponenten eines Hauses
$C = \{floor, wall_1, wall_2, wall_3, wall_4, ceiling, roof, paint_1, paint_2, paint_3, paint_4\}$
und eine Menge von Arbeitern $W = \{a, b, c\}$. Ein Beispielzustand wäre dann $<(floor, a), (wall1, b), (wall2, c)>$.
Die zeitlichen Abhängigkeiten der Komponenten sind wie folgt:

$$\begin{array}{lcl} a & \underset{\text{needs}}{\longrightarrow} & b \\ paint_n & \longrightarrow & wall_n \\ ceiling & \longrightarrow & wall_1 \wedge wall_2 \wedge wall_3 \wedge wall_4 \\ roof & \longrightarrow & ceiling \end{array}$$

Dadurch sind bestimmte Zustände oder Zustandsfolgen nicht möglich. Gesucht wird der kürzeste Pfad zum Ziel (alle Komponenten sind fertig gebaut), also ist die Breitensuche die beste Wahl.

- $state :< x, f_1, ..., f_n >$ wobei $f_n \in \mathbb{N}^0 \cup -1$ und n die Anzahl der Stockwerke ist. In jedem f_n ist die maximale Wartezeit der Personen codiert, -1 bedeutet dass niemand wartet. x ist das aktuelle Stockwerk des Fahrstuhls. Ein Zustandsübergang bewegt den Fahrstuhl um ein Stockwerk nach oben oder unten $(x + -1)$ und es vergeht eine Zeiteinheit $(\forall f_n : n \neq x : f_n + 1)$.
  Wenn $f_n : n = x \wedge f_n > 0$ nimmt der Fahrstuhl die wartenden Gäste mit und $f_n := -1$. Wenn ein Fahrgast im Fahrstuhl ist wählt er ein Stockwerk n zu dem er fahren will. Ist $f_n > 0$ dann bleibt es unverändert, ist es -1 wird es auf 0 gesetzt.

Der entsthende Graph ist bei immer wieder nachkommenden Fahrgästen unendlich. Das Ziel ist das Minimum aller f_n zu erreichen. Dafür kann die Breitensuche verwendet werden.