Das „Warehouseman's Problem“ ist PSPACE-schwer

Oliver Jelinski

12. Juli 2007

1 Einführung

Beweisziel: „Warehouseman's Problem“ oder auch das „Problem des Lagerarbeiters“, wie es wörtlich übersetzt werden könnte, ist PSPACE-schwer.

Problem des Lagerarbeiters: Eine Anzahl rechteckiger Objekte in einem zweidimensionalen rechteckigen Bereich soll aus einer Konfiguration in eine andere überführt werden durch Verschieben einzelner Objekte, wobei weder die Umrisse der anderen Rechtecke, noch der äußere Rahmen des Bereichs geschnitten werden dürfen.

2 Vorschau

Weg des Beweises: Ausgangspunkt ist *Rewrite-Problem für Ketten abstrakter Zeichen (1: ab hier RWP)*, von dem bekannt ist, dass es PSPACE-vollständig ist [2], dieses wird auf ein bestimmtes *Transpositions-Problem für Ketten abstrakter Zeichen* (2: ab hier TPP) reduziert, dieses dann wiederum auf ein *Verschiebe-Problem für bestimmte Objekte in einem rechteckigen zweidimensionalen Raum* (3: ab hier $2DM$), und dieses dann schließlich auf das *„Möbelrücken“ bzw. Problem des Lagerarbeiters* (4: ab hier WMP).

Es wird also die folgende Relation bewiesen:

$$RWP \leq_P TPP \leq_P 2DM \leq_P WMP$$

3 Grundlage: Ein PSPACE-vollständiges Rewriting-System

Grundlage für alle weiteren Reduktionen: Rewriting-System für eine Kette abstrakter Zeichen, dass $PSPACE$-vollständig ist:

Definition 1 *$PSPACE$-vollständiges Rewrite-System*
nach Hopcroft, Joseph und Whitesides [2]

1. *Es sei eine Zeichenkette $S = \{S_1 S_2 \cdots S_i \cdots S_m\}$ mit allen S_i aus einem Alphabet Σ*

2. *Es sei weiter eine Menge von Produktionen P_j, mit $0 < j \leq n$ die jeweils eine der folgenden Formen haben:*

   Typ $M := AB \rightarrow AC$

   Typ $N := AB \rightarrow CB$

   mit $A, B, C \in \Sigma$

3. *Es sei gewährleistet, dass auf jede Zeichenkette S', die durch sequentielle Produktionen aus S erzeugt wurde, jeweils genau zwei Produktionen angewandt werden können, und zwar genau eine vom Typ M und eine vom Typ N.*

4. *Schließlich sei gewährleistet, dass die jeweils anwendbaren Produktionen sich in mindestens einem Zeichen überschneiden, und dass in dem Fall, dass sie sich in* genau *einem Zeichen überschneiden, beide genau dieses Zeichen in ein anderes überführen.*

Wie eine solche Sprache aussähe, will ich hier nicht zeigen.

4 Reduktionen

4.1 Reduktion auf ein Verschiebe-Pproblem für Zeichenketten (TPP)

Ausgangskonfiguration

enthält:

1. - nicht nur die anfängliche Zeichenkette S des RWP
   - sondern ist in die *Klammerzeichen* Λ und Γ eingeschlossen, ab hier der *signifikante Teil*
   - rechts davon noch alle weiteren Zeichen aus Σ, ab hier *Standardzeichen* genau n mal, abzüglich der Anzahl seiner Instanzen innerhalb von S

2. Beim Verschieben darf in jedem Schritt nur ein Zeichen die Position wechseln, da auch in RWP jeweils nur ein Zeichen verändert wird.

3. Klammerzeichen dürfen nicht die Position wechseln.

4. Außer bestimmten erlaubten Transpositionen (und zwar genau zwei pro Schritt) müssen alle Verschiebungen verboten sein. Es sei dafür (für die Konstruktion dieses Verbots)

   - jedem Standardzeichen ein Index 0, 1 oder 2 zugewiesen
   - die Regel festgelegt, dass jedes Standardzeichen σ_i nur rechts von einem Standardzeichen stehen darf, wenn dieses den Index $i - 1 \bmod 3$ hat, und nur links von einem Standardzeichen, wenn dieses den Index $i + 1 \bmod 3$ hat.

   Damit bisher *keine* Möglichkeit, den signifikanten Teil der Zeichenkette zu verändern.

   Alphabet so *verdreifacht*, weil aus jedem Symbol σ die drei Symbole σ_0, σ_1 und σ_2 geworden sind.

   In dem Teil von S_{TPP} rechts des signifikanten Teils jetzt alle Symbole σ_i in ausreichender Zahl gespeichert.

5. Problem: Nachbarschaftsregeln untersagen, dass Zeichen mit dem gleichen Index neben einander stehen.

   Lösung:

   - (Nicht-Klammer-)Zeichen im nichtsignifikanten Teil von zwei Klammerzeichen '[' und ']' umschlossen
   - für Standardzeichen, die im signifikaten Teil stehen, leere Klammerpaare '[]'vorhanden

6. Problem: keine einzige Verschiebeaktion mehr erlaubt.

   Lösung: *Spezialzeichen*

   - für jede Produktion P_i aus RWP drei Spezialzeichen
   - M^i_{01}, M^i_{12} und M^i_{20} wenn $AB \to AC$ (also Typ M)
   - N^i_{01}, N^i_{12} und N^i_{20}, wenn $AB \to CB$ (also Typ N)
   - werden zunächst zwischen Klammern '[' und ']' im nicht signifikanten Teil gespeichert.

Die Zeichenkette hat so die folgende Gestalt:

$$\Lambda S_{1_0} S_{2_1} S_{3_2} S_{4_0} S_{5_1} S_{6_2} S_{7_0} \cdots \Gamma$$

$$\cdots [M^i_{01}][M^i_{12}][M^i_{20}] \cdots [M^j_{01}][M^j_{12}][M^j_{20}] \cdots$$

$$\cdots [N^k_{01}][N^k_{12}][N^k_{20}] \cdots [N^l_{01}][N^l_{12}][N^l_{20}] \cdots$$

$$[\sigma_{1_0}][\sigma_{1_0}] \cdots [\sigma_{1_0}][][] \cdots [][\sigma_{1_1}][\sigma_{1_1}] \cdots [\sigma_{1_1}][][] \cdots [][\sigma_{1_2}][\sigma_{1_2}] \cdots [\sigma_{1_2}][][] \cdots []$$

$$\vdots$$

$$[\sigma_{m_0}][\sigma_{m_0}] \cdots [\sigma_{m_0}][][] \cdots [][\sigma_{m_1}][\sigma_{m_1}] \cdots [\sigma_{m_1}][][] \cdots [][\sigma_{m_2}][\sigma_{m_2}] \cdots [\sigma_{m_2}][][] \cdots []$$

Nachbarschaftsregeln M, N

- Jedes M^i_{jk} darf zwischen zwei Standardzeichen stehen, genau dann, wenn die folgenden beiden Bedingungen zutreffen:
  - Das links angrenzende Zeichen entspricht dem ersten Zeichen in der Produktionsregel P_i *und* der Index des Zeichens ist gleich j.
  - Das rechts angrenzende Zeichen hat *entweder* den Index k und entspricht dem zweiten Zeichen der rechten oder der linken Seite der Produktionsregel P_i, *oder*, es hat den Index $k + 1 \bmod 3$.
- Jedes N^i_{jk} darf zwischen zwei Standardzeichen stehen, genau dann, wenn die folgenden beiden Bedingungen zutreffen:
  - Das rechts angrenzende Zeichen entspricht dem zweiten Zeichen in der Produktionsregel P_i *und* der Index des Zeichens ist gleich k.
  - Das links angrenzende Zeichen hat *entweder* den Index j und entspricht dem ersten Zeichen der rechten oder der linken Seite der Produktionsregel P_i, *oder*, es hat den Index $j - 1 \bmod 3$.
- Kein M^i_{jk} oder N^i_{jk} darf direkt neben einem weiteren Spezialzeichen stehen.

Ergebnis: $RWP \leq_P TPP$

„⇒ “: Jede Produktion P_i der Form $AB \to AC$ aus RWP kann simuliert werden durch folgende Verschiebesequenz:

$$\Lambda \cdots ABE \cdots \Gamma \cdots [\mathbf{M^i}] \cdots [] \cdots [C] \cdots$$

$$\Rightarrow \Lambda \cdots AM^i\mathbf{B}E \cdots \Gamma \cdots [] \cdots [] \cdots [C] \cdots$$

$$\Rightarrow \Lambda \cdots AM^iE \cdots \Gamma \cdots [] \cdots [B] \cdots [\mathbf{C}] \cdots$$

$$\Rightarrow \Lambda \cdots A\mathbf{M^i}CE \cdots \Gamma \cdots [] \cdots [B] \cdots [] \cdots$$

$$\Rightarrow \Lambda \cdots ACE \cdots \Gamma \cdots [M^i] \cdots [B] \cdots [] \cdots \quad \surd$$

Fett gedruckt ist jeweils das Zeichen, das als nächstes verschoben wird.

„⇐ “: *nur* jeweils M^i und N^j, die den Produktionen P_i und P_j entsprechen, dürften verschoben werden. Es muss sichergestellt werden, dass hier auch nur *genau* eine der beiden Verschiebesequenzen möglich ist, die oben dargestellt würden.

1. Fall: Linke Seiten der Produktionen P_i überlappen in genau einem Zeichen. Wenn P_i die Form $AB \to AC$ hat, muss dann P_j die Form $BE \to DE$ haben. Auf das Verschieben von M^i kann das Verschieben von N^j folgen:

   $$\Rightarrow \Lambda \cdots AM^iBE \cdots \Gamma \cdots [] \cdots [\mathbf{N^j}] \cdots [] \cdots [C] \cdots$$

   $$\Rightarrow \Lambda \cdots AM^iBN^jE \cdots \Gamma \cdots [] \cdots [] \cdots [] \cdots [C] \cdots$$

   Aber: kein Progress mehr möglich. B darf nicht entfernt werden, weil sonst zwei Spezialzeichen nebeneinander stünden. Da auch A und E nicht entfernt werden dürfen (was aus den zugehörigen Produktionen klar wird), einzige Möglichkeit: entweder M^i oder N^j wieder entfernen. $\surd$

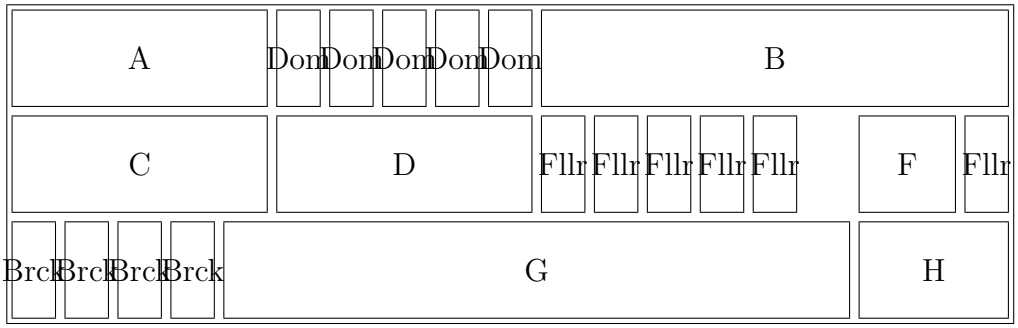
2. Fall: Linke Seiten überlappen in beiden Zeichen. Dann müssten M^i und N^j an dieselbe Stelle gesetzt werden. Aber: Einsetzen eines Spezialzeichens neben ein anderes nicht erlaubt. $\surd$

Jede Instanz von RWP ist also genau dann erfüllbar, wenn die entsprechende Instanz für TPP erfüllbar ist. RWP ist so auf TPP reduziert. Insgesamt dürfte deutlich sein, dass die Reduktion in polynomieller Zeit erfolgen kann. TPP ist also $PSPACE$-schwer.

4.2 Reduktion auf ein Verschiebe-Problem für Objekte im 2-dimensionalen Raum ($2DO$)

Ausgangssituation (vorläufig)

Es muss eine Reihe von zweidimensionalen Objekten im rechteckigen Raum geben, von denen jeweils *genau* eines aus der Reihe heraus genommen und an anderer Stelle wieder eingefügt werden kann. Das leistet folgende Konstruktion:



Verschiebe-Sequenzen (vorläufig)

1. Erste Verschiebesequenz

   - „Fllr“, also Füllblöcke nach rechts
   - Block D und danach C hinterher
   - links dadurch neue Lücke
   - von unten „Brck“, also so etwas wie ein ein Pflasterstein, in die Lücke
   - andere Pflastersteine und den Block G nach links
   - schon verschobenen Füllblock nach unten

   Damit ist der anfängliche Freiraum wieder hergestellt. Man könnte sagen, eine „Runde“ ist vorbei.

   - alle Bewegungen, die am Anfang möglich waren, wieder möglich, obwohl sich die Konstellation verändert hat
   - Runde lässt sich so oft wiederholen, wie Pflastersteine am linken Rand nach oben nachgeschoben werden können, in der schematischen Darstellung viermal
   - jede dieser Runden kann auch rückwärts gemacht werden

- Entscheidend: mit jeder „Runde“, vorwärts oder rückwärts, verschiebt sich die Position der Lücke zwischen den Blöcken C und D. Entscheidend, weil diese Lücke in der zweiten Bewegung erweitert und mit einem der Dominosteine gefüllt wird – mit welchem, hängt davon ab, wie weit die Lücke zwischen C und D bereits bewegt wurde.

2. zweite Verschiebesequenz

   - alle zusammenhängenden Füllblöcke neben dem Block D nach rechts
   - Block D gleich hinterher
   - Block C bleibt an seinem Platz
   - einer der „Dominosteine“ in die Lücke zwischen C und D
   - Dominosteine rechts von diesem nach links in die entstandene Lücke
   - Block B hinterher
   - einzelner Füllstein rechts von Block F in die über ihm entstandene Lücke
   - Block E und links von ihm befindlicher Füllstein ganz nach rechts.

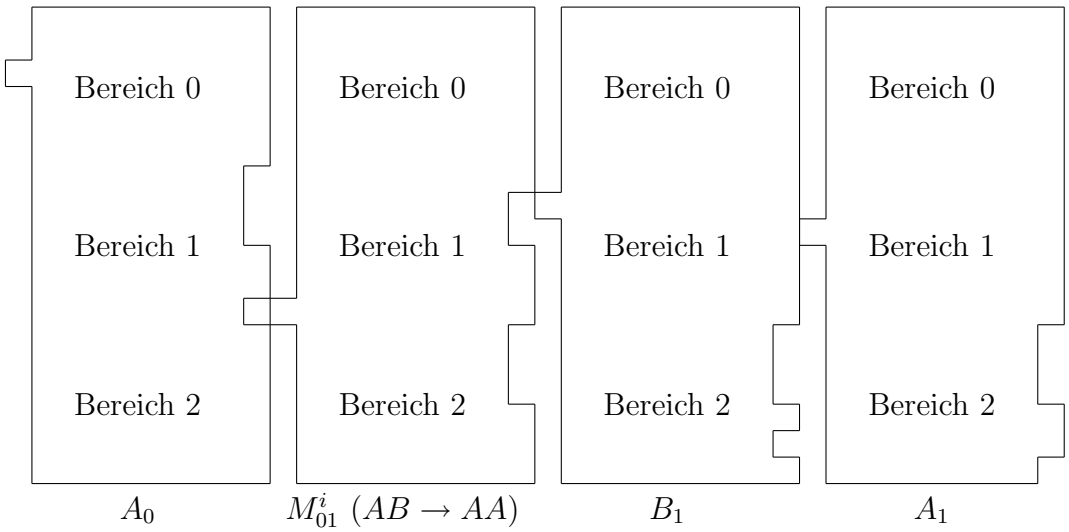
   Auch hier ist die Leerstelle jetzt wieder an dem Platz, an dem sie am Anfang der Bewegung war. Man könnte auch hier davon reden, dass eine „Runde“ vorbei ist.

   - diese Bewegung nicht wiederholbar, sondern nur rückgängig zu machen

Zusammenspiel: jeder der Dominosteine kann aus der Reihe herausgezogen, transportiert und wieder eingesetzt werden kann. Die „Transportmaschine“ simuliert also ein Verschiebeproblem – allerdings bisher nicht das, was simuliert werden soll, denn bisher kann jeder Dominostein neben jeden wieder eingesetzt werden. Es gibt *noch keine Nachbarschaftsregeln.*

Simulation der Nachbarschaftsregeln

: Dominosteine modifizieren. Jeder bekommt, je nachdem, ob er ein Standardzeichen, ein Spezialzeichen, oder ein Klammerzeichen ist, und jenach dem, welches, bestimmte spezifische Ausbuchtungen, die nur in die spezifischen Einbuchtungen bestimmter Steine geschoben werden können, und spezifische Einbuchtungen, in die nur die Ausbuchtungen bestimmter anderer Steine geschoben werden können.



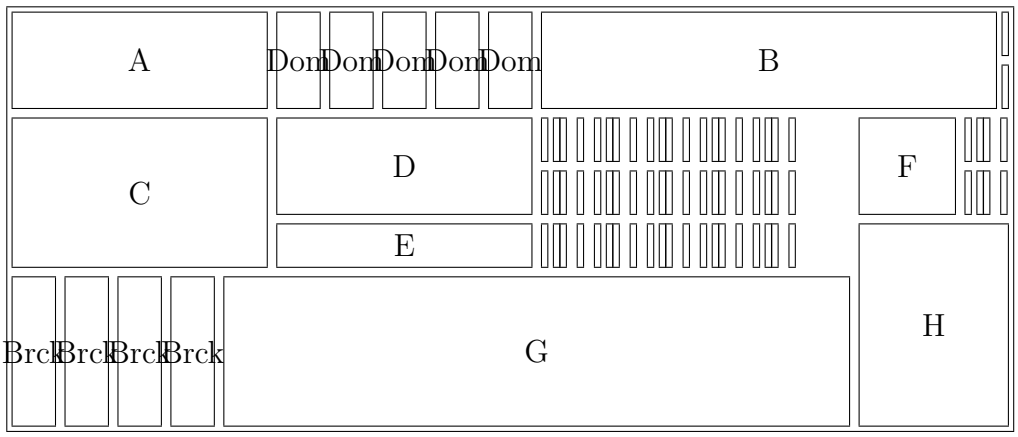
Man sieht leicht, dass sowohl der A_1 repräsentierende, als auch der B_1 repräsentierende Stein an den M_{01}^i repräsentierenden passen. Auch jeder Stein, der ein Standardzeichen mit dem Index 2 repräsentierte, würde rechts passen.

Wie genau das geht, zeige ich hier nicht. Entscheidend: Damit sind alle Nachbarschaftsregeln erfüllt.

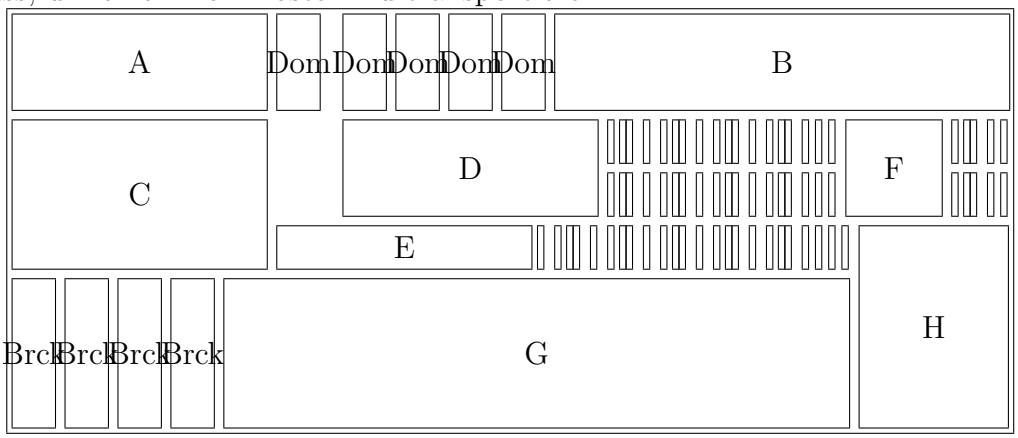
Problem, Modifikation

Problem: Steine jetzt zu breit, passen nicht mehr in die Lücke zwischen Block C und D und sind so verkeilt, dass sie auch nicht nach unten verschoben werden können.

Lösung: Zwischenräume ein kleines Stück weiter öffnen. Modifikation der Transportmaschine:



Durch geschickte Umordnung und Teilung der Füllblöcke kann so der Platz geschaffen werden, der durch die Ausbuchtungen zusätzlich freigemacht werden muss, um einen Dominostein zu transportieren:



Ergebnis: $TPP \leq_P 2DO$

Damit simuliert $2DO$ genau TPP, und eine Instanz TPP ist genau dann erfüllbar, wenn die zugehörige Instanz von $2DO$ erfüllbar ist. Damit ist TPP auf $2DO$ reduziert, und dass das ganze in polynomieller Zeit möglich ist, dürfte auch offensichtlich sein.

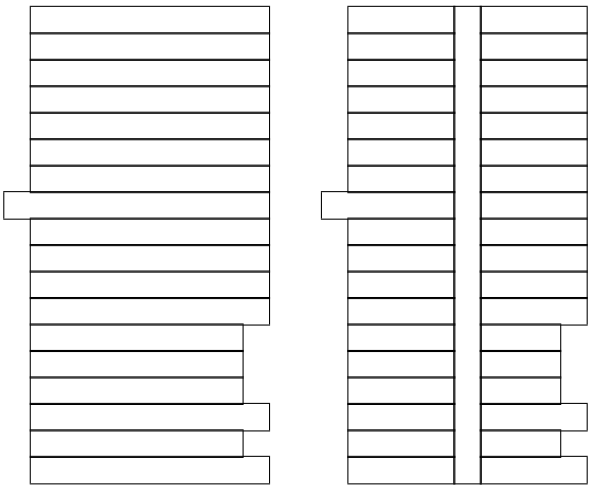
$2DO$ ist also $PSPACE$-schwer.

4.3 Reduktion auf das Problem des Lagerarbeiters (WMP)

Problem: Dominosteine nicht rechteckig.

Idee zur Lösung: Dominosteine horizontal zerteilen.

Aber: ungewollte Verschiebungen möglich. Dagegen: „Rückgrad“ zwischen die „Rippen“:



Weitere Probleme:

- Vermischung von Teilen zwischen Dominosteinen
- Vermischung von Teilen eines Dominosteins

Voraussetzung der Lösung:

- Zwischenblöcke, doppelt so breit wie normale Dominosteine.
- Zwischenlagen zwischen „Rippen“ , die in Zwischenblöcken weniger breit sind, in Dominosteinen breiter, so dass zwei Dominosteine nich direkt nebeneinander passen.

1. Problem: Vermischung von Teilen eines Dominosteins

   Lösung: Die verschiedenen Lagen eines Dominosteins nicht gleichhoch, sondern, wenn die unterste Lage (Lage 0) die Höhe h hat, sei die i-te Lage $h/3^i$ hoch.

   Da jede Lage i höher ist als die Summe aller Lagen $> i$, kann hier nichts mehr vertauscht werden.

2. Problem: Vermischung von Teilen zwischen Dominosteinen

   Lösung:

- Höhen der Steine von den Höhen der Lagen minimal unterschieden
- Steine der signifikanten Lage i auf der linken Seite um δ_{Li} höher sind als die Lage, und die der darüberliegenden nicht signifikanten Lage (Trennlage) um $\delta_{L(i+1)}$ weniger hoch als die Lage
- δ_{Rj} so verteilt, dass die Steine der signifikanten Lage j auf der rechten Seite um δ_{Rj} höher sind als die Lage, und die darüberliegenden der nicht signifikanten Trennlage um $\delta_{R(j+1)}$ weniger hoch
- $\Sigma\delta_{Li} = 0$ und $\Sigma\delta_{Rj} = 0$
- Unterschiede so gewählt, dass es keine Summe von mehreren δ_i und/oder δ_j gibt, die gleich irgendeiner anderen Summe anderer δ_i und/oder δ_j ist.

Höhe der Steine auf der rechten und linken Seite jeweils gleich der Höhe aller Lagen. Jede Vertauschung von Steinen würde dazu führen, dass entweder die Steine auf der linken oder die auf der rechten Seite höher würden als die Summe der Lagen, wodurch sie nicht in die Transportlücke passen würden. So können auch keine rechten und linken Steine von Dominosteinen vertauscht werden.

Zwei Zwischenblöcke nebeneinander, um zu ermöglichen, einen Dominostein einzufügen, der ein Spezialzeichen repräsentiert.

Ergebnis: $2DO \leq_P WMP$

Damit ist die Reduktion abgeschlossen. (Die Autoren benötigen ein paar weitere Modifikationen; warum ist mir noch nicht klar geworden.) Dass die Reduktion in polynomieller Zeit von Statten geht, dürfte auch hier klar sein.

Das Problem des Lagerarbeiters ist also $PSPACE$-schwer.

Literatur

[1] J.E. Hopcroft, J.T. Schwartz, M. Sharir, „On the Complexity of Motion Planning for Multiple Independent Objects; PSPACE-Hardness of the ≪Warehouseman's Problem≫“, in: The International Journal of Robotic Research, 1984

[2] J.E. Hopcroft, D. Joseph, S. Whitesides, „Movement problems for 2-dimensional Linkages“, SIAM Journal Computing, 1984 (erstmalig 1982)

[3] Ingo Wegener, Theoretische Informatik - eine algorithmenorientierte Einführung, Stuttgart/Leipzig, 1999