Einführung in algorithmische Geomterie

Arne Schmidt

Heute

Was passiert, wenn mehrere Roboter existieren?

Die sollten natürlich nicht kollidieren...

Wie schwer ist es, ein Bewegungsplan zu finden?

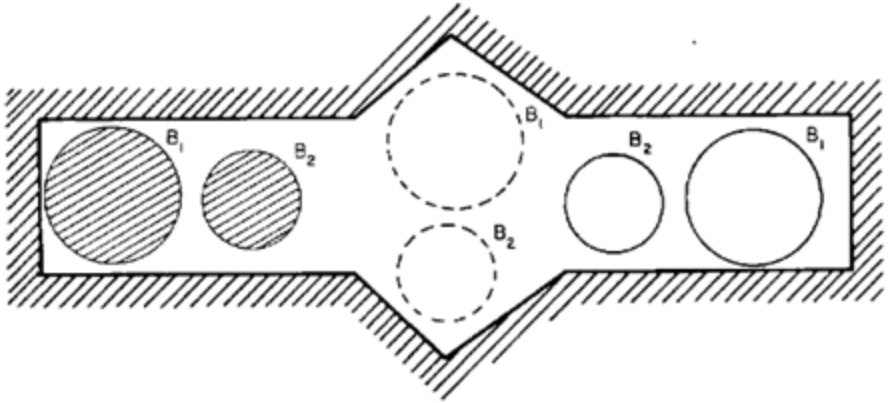
Kapitel 3.5 – Multi-Robot-Motionplanning

Das Problem

Jacob T. Schwartz
Computer Science Department
Courant Institute of Mathematical Sciences
New York University
New York, New York 10012

Micha Sharir
School of Mathematical Sciences
Tel Aviv University
Tel Aviv, Israel

On the Piano Movers' Problem: III. Coordinating the Motion of Several Independent Bodies: The Special Case of Circular Bodies Moving Amidst Polygonal Barriers



Polynomialzeit-Algorithmus bzgl. Komplexität der Umgebung, aber exponentiell in Anzahl Roboter.

„Vereinfachung“

Kreise sind geometrisch manchmal schwierig. Nehmen wir zunächst **Quadrate** an!

Unterschiedliche Größen sind nervig. Betrachten wir **Einheitsquadrate**!

Unterscheidbare Roboter erzeugen viele kreuzende Pfade. Nehmen wir die **unlabeled** Variante!

Das Problem

Gegeben:
- Ein Workspace W,
- n Einheitsquadrate mit Startpositionen $s_1, \dots, s_n$ in W
- Zielpositionen $g_1, \dots, g_n$

Frage:
Können die Einheitsquadrate von den Startpositionen zu den Zielpositionen bewegt werden, sodass...
- Am Ende jede Zielposition von einem Quadrat besetzt ist.
- Zu keinem Zeitpunkt zwei Quadrate sich überlappen.
- Zu keinem Zeitpunkt ein Quadrat sich mit dem Forbidden Space überlappt.

PSPACE

Theorem:
Das genannte Problem ist PSPACE-vollständig

On the hardness of unlabeled multi-robot motion planning*

Kiril Solovey and Dan Halperin
Blavatnic School of Computer Science
Tel Aviv University, Israel
email: {kirilsol,danha}@post.tau.ac.il

PSPACE

Definition:

a) Ein (Entscheidungs-)Problem P liegt in der Klasse **PSPACE**, wenn es einen Algorithmus gibt, der P mit polynomiellen Speicherplatzbedarf löst.

b) Ein (Entscheidungs-)Problem P heißt **PSPACE-schwer**, wenn jedes Problem aus PSPACE in polynomieller Zeit auf P reduziert werden kann.

c) Ein (Entscheidungs-)Problem P heißt **PSPACE-vollständig**, wenn P sowohl in PSPACE liegt, als auch PSPACE-schwer ist.

Das „Master-Problem“ für PSPACE ist Quantified Satisfiability:

$$\exists x_1 : \forall x_2 : \exists x_3 : \forall x_4 : \ldots : \varphi(x_1, \ldots, x_n) = true$$

Contraint Graphen

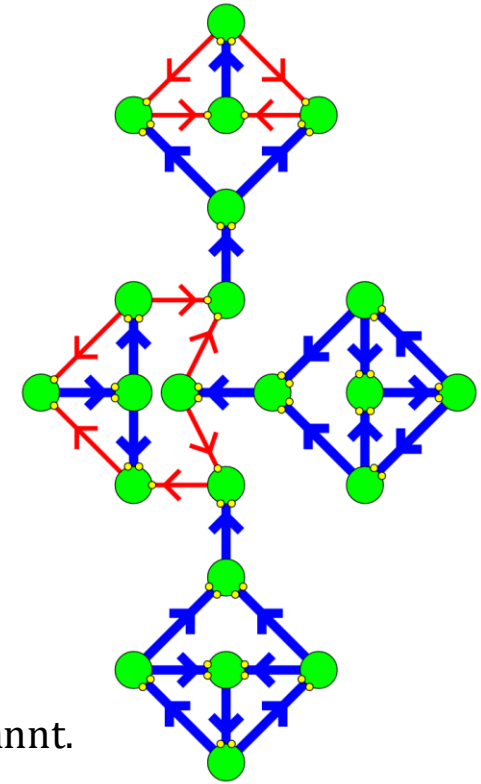
Definition:
Sei $G = (V, E)$ ein ungerichteter Graph mit Kantengewichten $c: E \to \{1, 2\}$, in welchem jeder Knoten

- zu genau 3 Kanten inzident ist.
- gerade viele inzidente Kanten mit Gewicht 1 besitzt.

G heißt auch and/or-constraint-Graph

Eine Konfiguration von G ist eine Orientierung der Kanten, sodass an jedem Knoten:

- Mindestens eine Kante von Gewicht 2 hineingeht, oder
- Mindestens zwei Kanten von Gewicht 1 hineingehen.



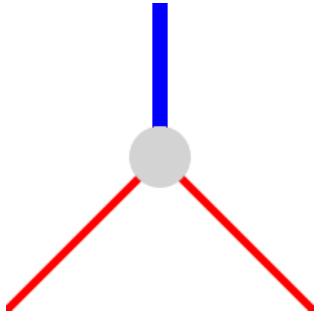
Kanten von Gewicht 1 (bzw. 2) werden auch rote (blaue) Kanten genannt.

Constraint Logic Problem

Problem:
Gegeben: Zwei Konfigurationen C_1, C_2 zu einem Constraint-Graphen G.
Frage: Gibt es eine Sequenz von Kantenumdrehungen, sodass aus C_1die Konfiguration C_2 wird?

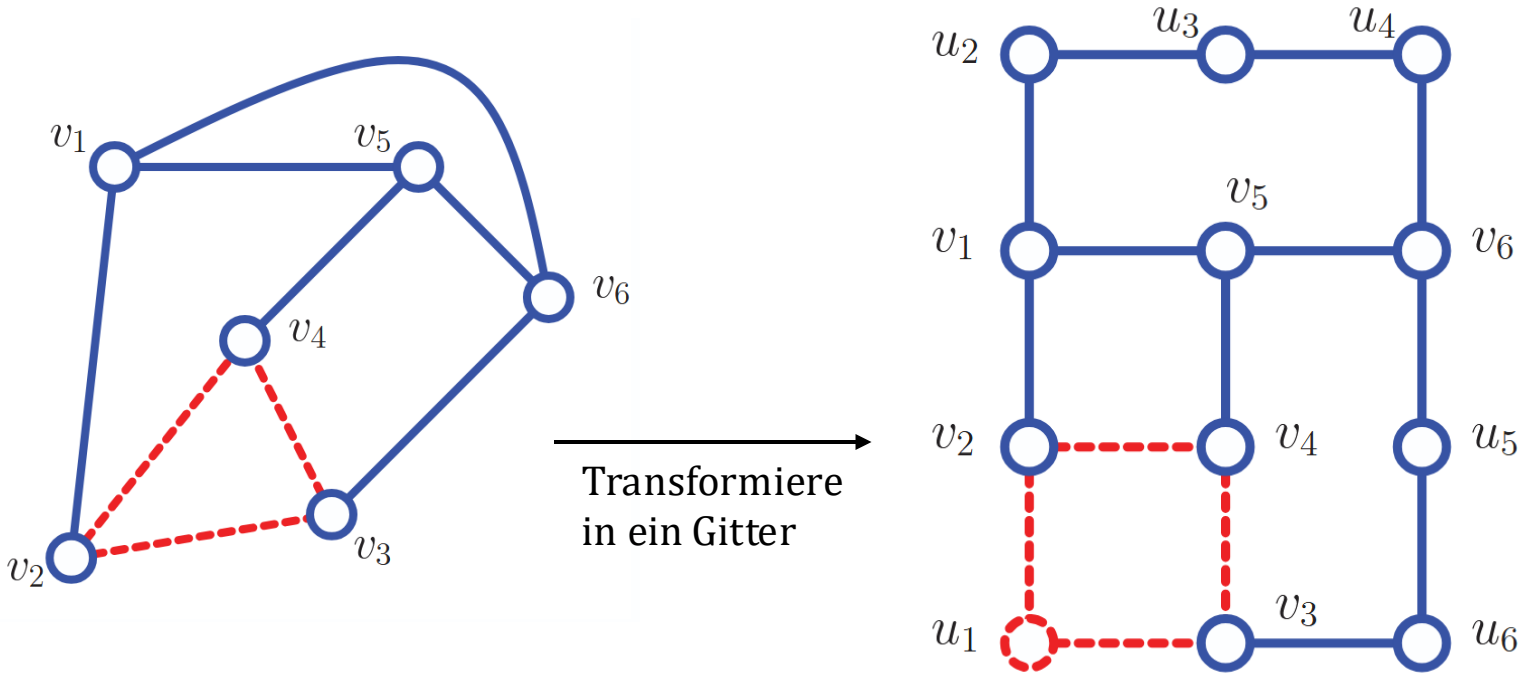
Theorem:
Dieses Problem ist PSPACE-vollständig.

Theorem:
Das Problem bleibt PSPACE-vollständig, wenn G ein planarer Graph ist, d.h. er kann ohne kreuzende Kanten gezeichnet werden.

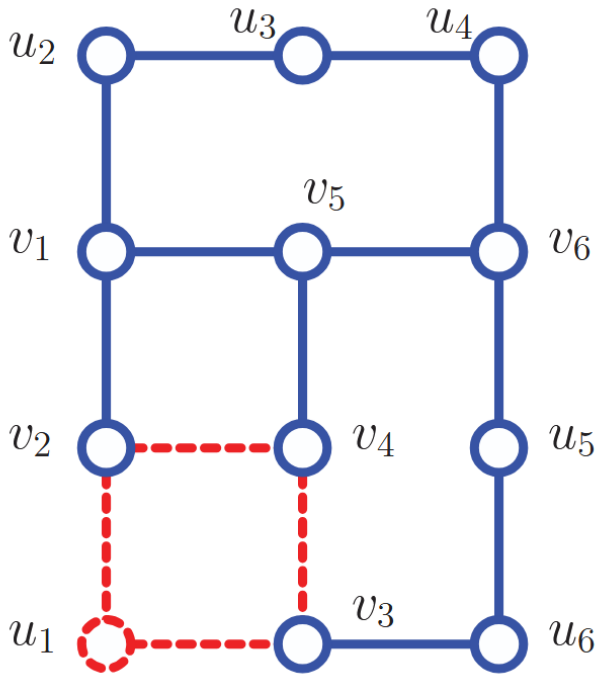


Theorem:
Das Problem bleibt PSPACE-vollständig, wenn für Knoten nur Gewicht 1 hineinfließen muss, sobald nur zwei Kanten anliegen.

Reduktion von CLP auf Robot Motion Planning



Reduktion von CLP auf Robot Motion Planning

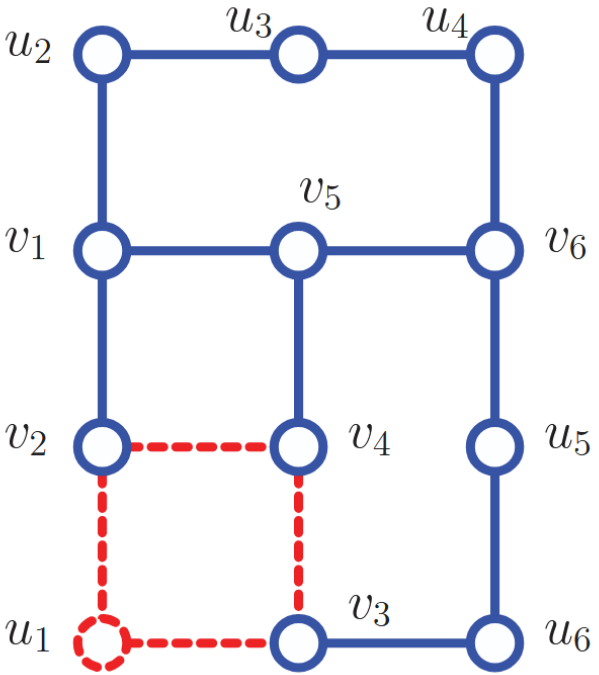


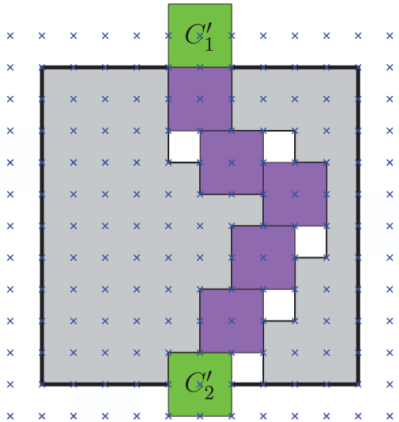
Jeder Knoten muss nun mit einem Gadget ersetzt werden, welche mit Quadraten gefüllt werden.

Wir benötigen:

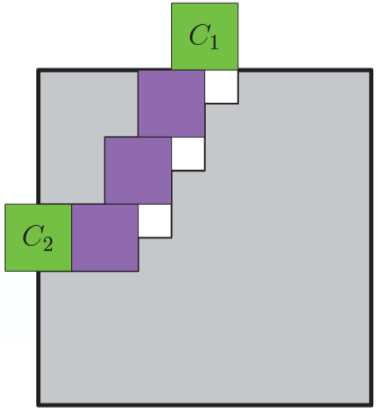
- Connector Gadgets für u_i
- And-Gadget für v_2, v_3, v_4
- Or-Gadget v_1, v_5, v_6

Connector-Gadgets

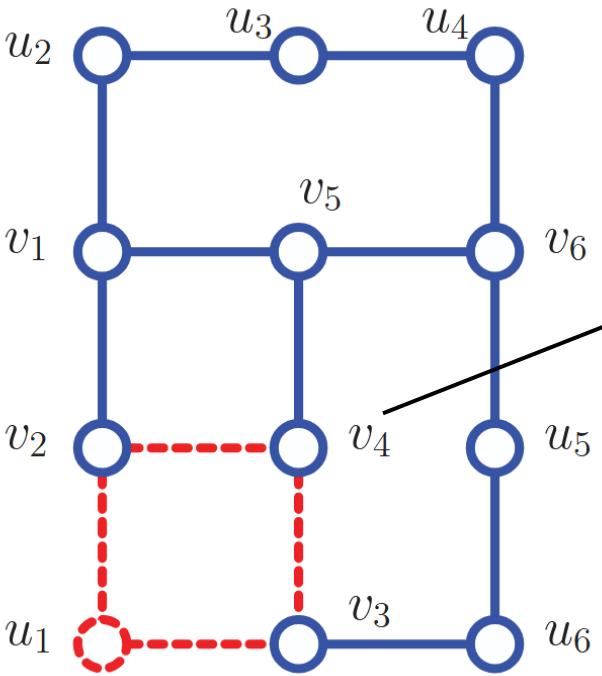


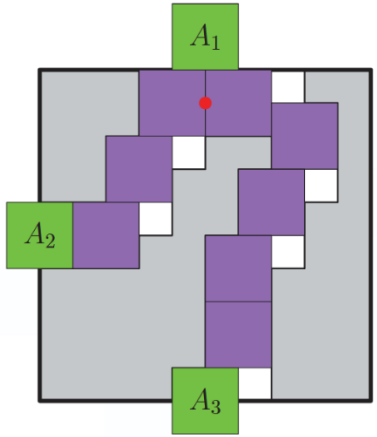


Richtung der Kante wird darüber entschieden, welches grüne Quadrat in die graue Box ragt.

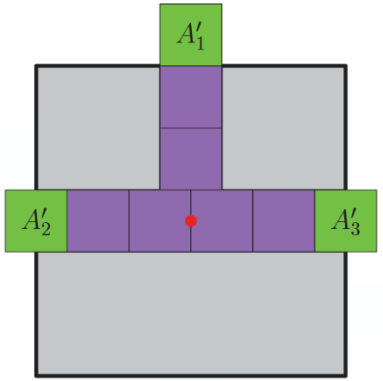


And-Gadgets



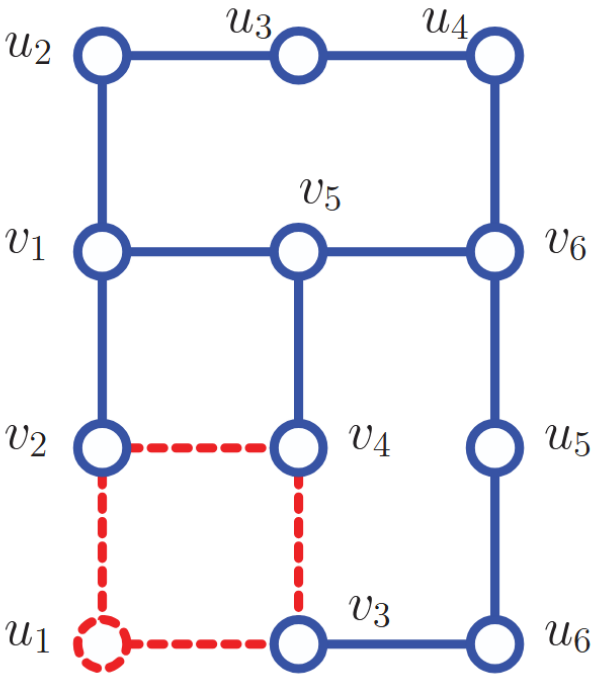


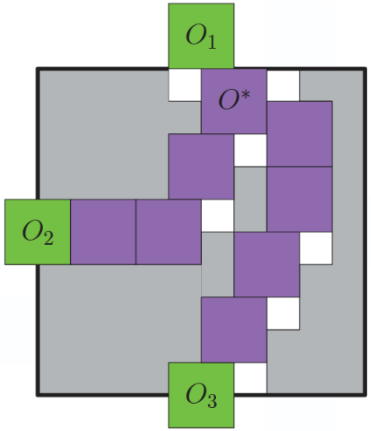
A_1 kann nur dann hineingehen, wenn A_2 und A_3 herausgehen!



A'_1 kann nur dann hineingehen, wenn A'_2 und A'_3 herausgehen!

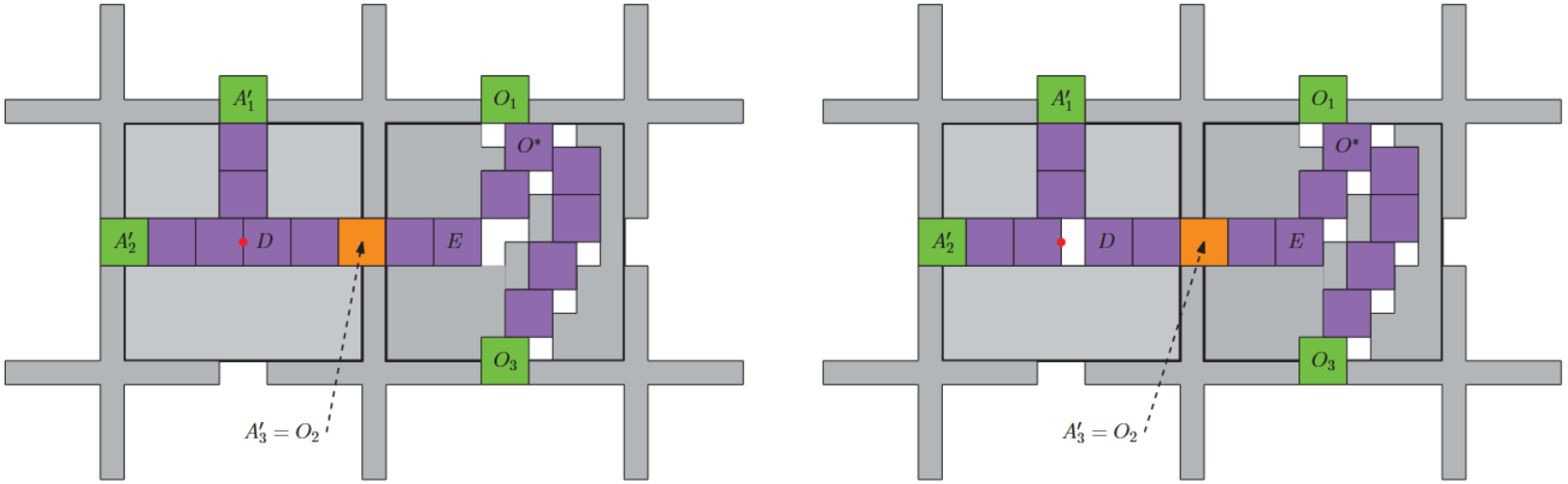
Or-Gadgets



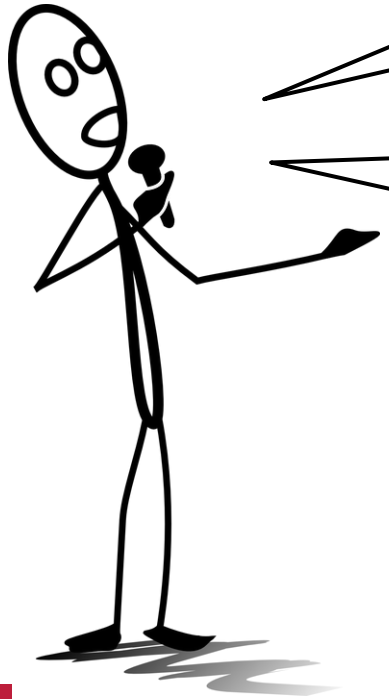


Die drei O_i können nicht gleichzeitig hineinragen. Mind. einer muss raus!

Beispiel für zwei verbundene Gadgets



Varianten



Das Problem bleibt schwer, auch wenn man Kreis-Roboter betrachtet.

Das Problem bleibt schwer, auch wenn man gelabelte Roboter betrachtet (jeder Roboter hat sein eigenes Ziel).

3.6 – „Einfacheres“ Multi-Robot Motion Planning

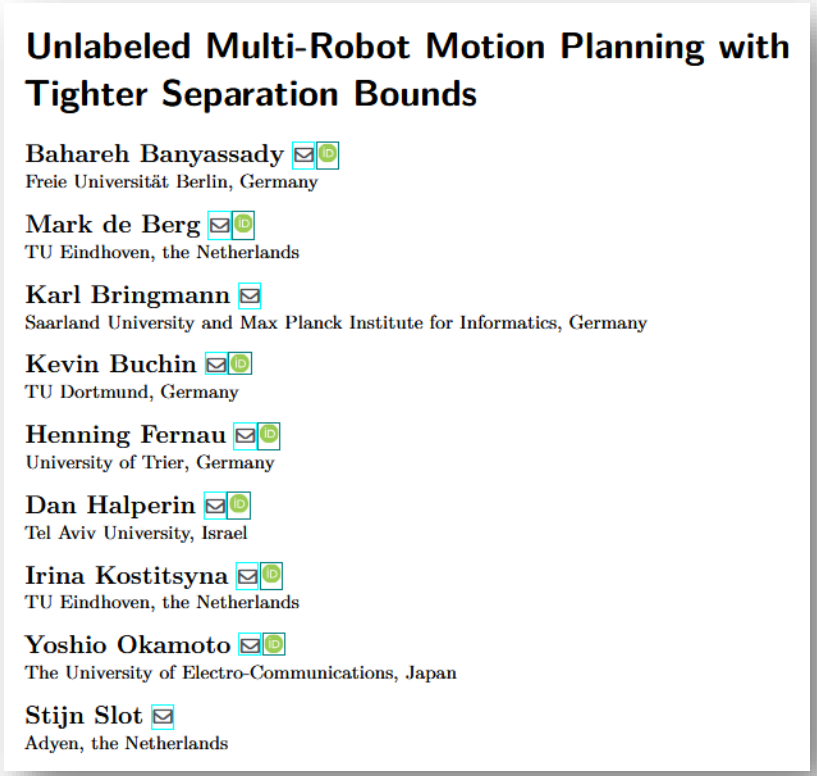
Überlegung 1

Gibt es bestimmte Annahmen, bei welchen das Problem einfacher wird?

Beobachtung: In der Reduktion sind alle Roboter sehr dicht gepackt...

Wie weit müssen Start- und Zielpositionen auseinander liegen, damit es immer eine Lösung gibt?

Separation Bounds



Unlabeled Multi-Robot Motion Planning with Tighter Separation Bounds

Bahareh Banyassady
Freie Universität Berlin, Germany

Mark de Berg
TU Eindhoven, the Netherlands

Karl Bringmann
Saarland University and Max Planck Institute for Informatics, Germany

Kevin Buchin
TU Dortmund, Germany

Henning Fernau
University of Trier, Germany

Dan Halperin
Tel Aviv University, Israel

Irina Kostitsyna
TU Eindhoven, the Netherlands

Yoshio Okamoto
The University of Electro-Communications, Japan

Stijn Slot
Adyen, the Netherlands

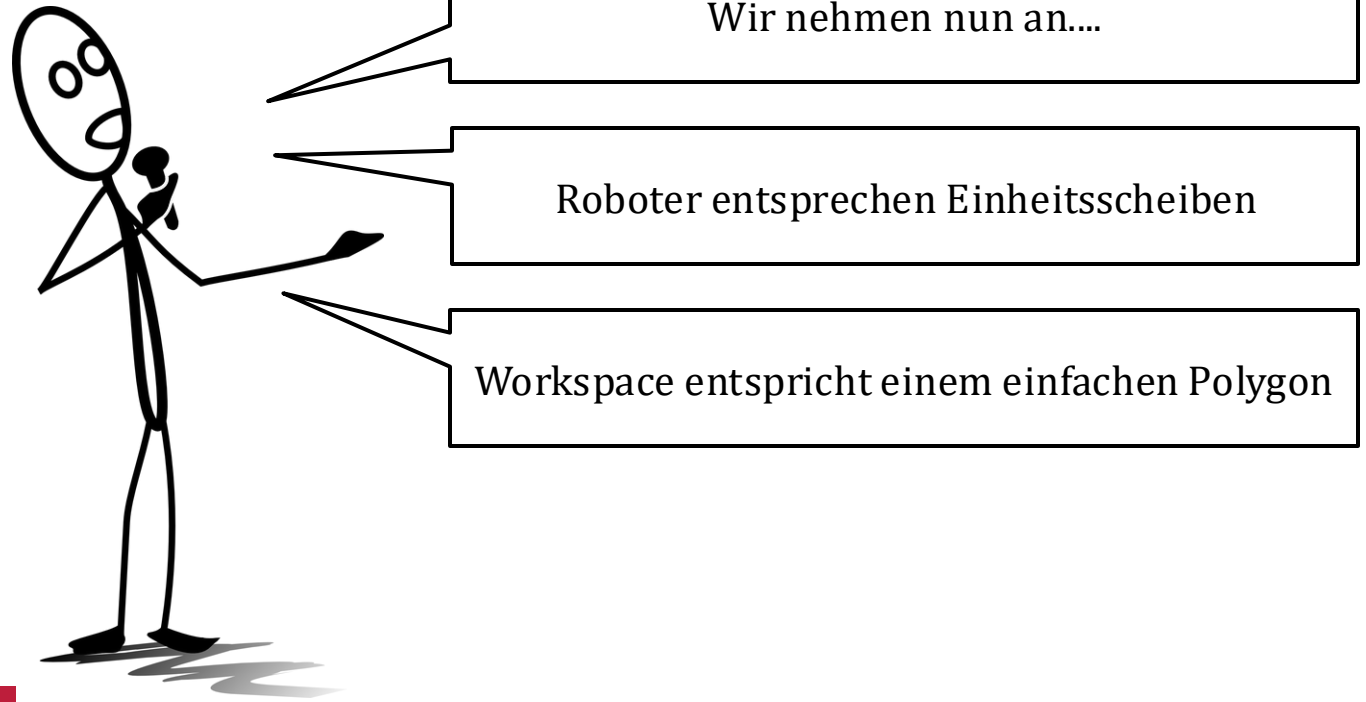
Sei μ der kleinste Abstand zwischen zwei Startpositionen, bzw. zwei Zielpositionen

Sei β der kleinste Abstand zwischen Start- und Zielpositionen.

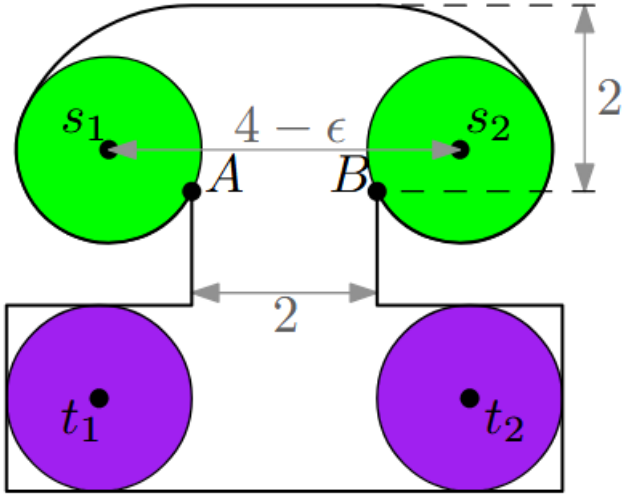
Grundfrage: Was ist das kleinste μ_0 und β_0, sodass

- Es gibt Instanzen, die keine Lösung mit $\mu < \mu_0$ besitzen.
- Es gibt Instanzen, die keine Lösung mit $\beta < \beta_0$ besitzen.
- Ab $\mu \geq \mu_0$ und $\beta \geq \beta_0$ ist es immer lösbar.

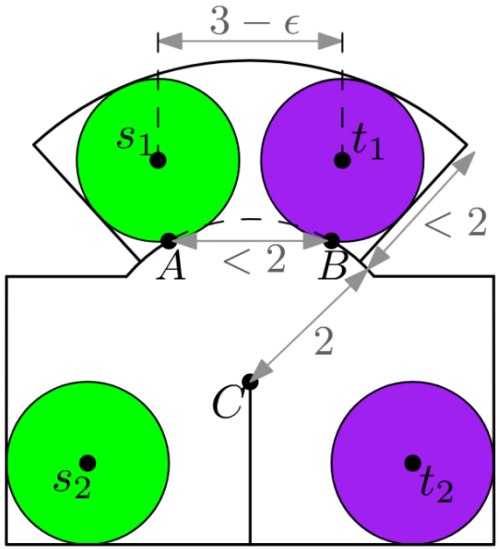
Annahmen


Wir nehmen nun an....
Roboter entsprechen Einheitsscheiben
Workspace entspricht einem einfachen Polygon

$\mu_0 \geq 4$

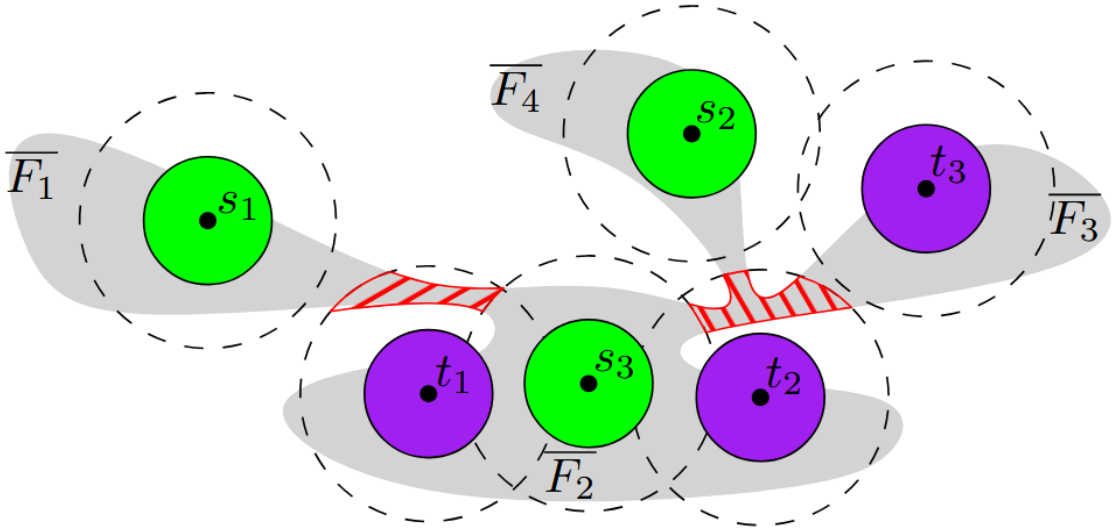


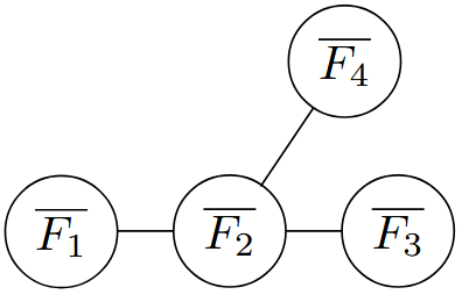
$\beta_0 \geq 3$



Hier ist der Freespace unzusammenhängend:
s_1 kann nicht auf t_2, s_2 kann nicht auf t_1 platziert werden.

Generelle Idee – Residualgraph des Freespace



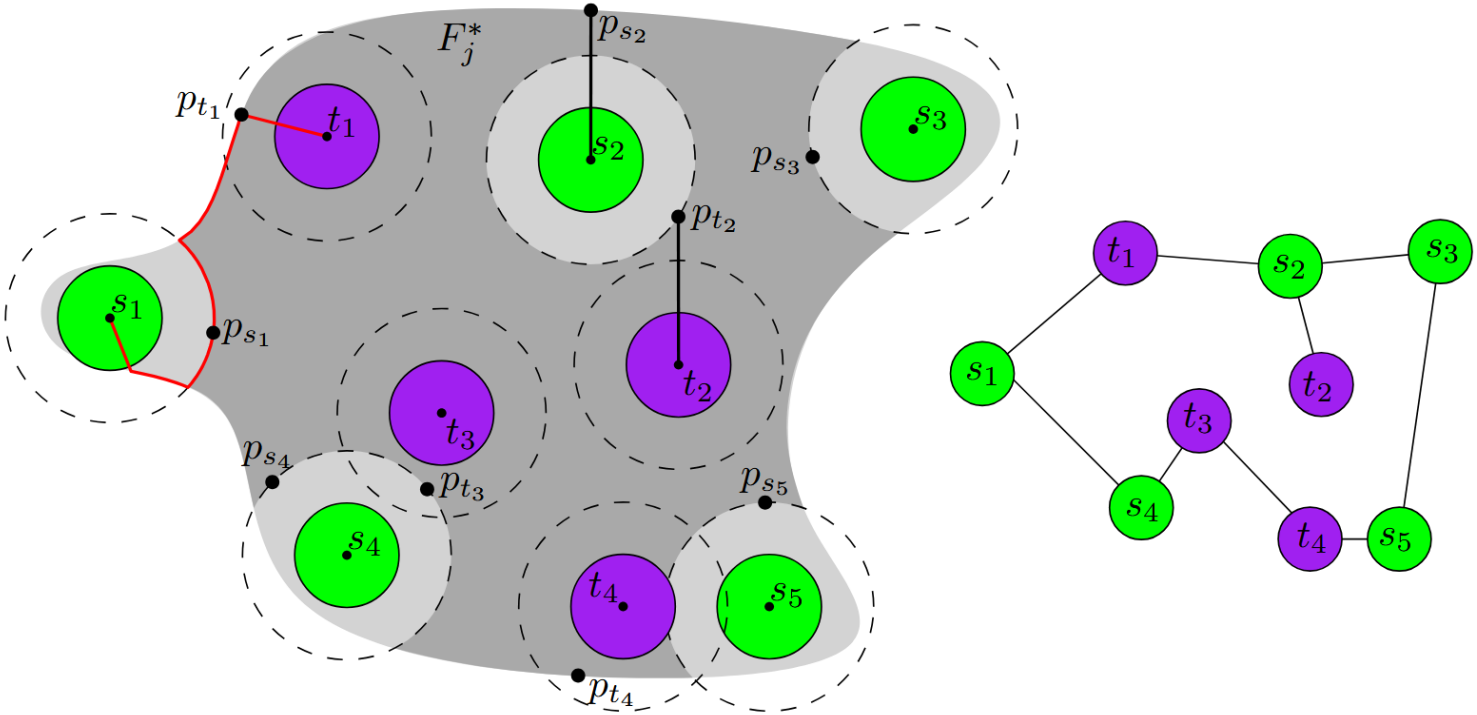


Roboter auf Zielpositionen blockieren die rote Fläche.

→ Müssen freibleiben, falls Roboter in andere Komponenten verschoben werden müssen!

→ Löse nach und nach die Blätter

Lösen einer Komponente



Zusammenhängender Freespace

Theorem:
Ist der Freespace *zusammenhängend*, dann kann eine Lösung des *ungelabelten* Motion Planning Problems für *Kreisroboter* in *einfachen Workspaces* mit $\mu \geq 4$ in Zeit $O(n \log n + mn + m^2)$ Zeit gefunden werden.

Achtung:
Hier gibt es viele Annahmen!

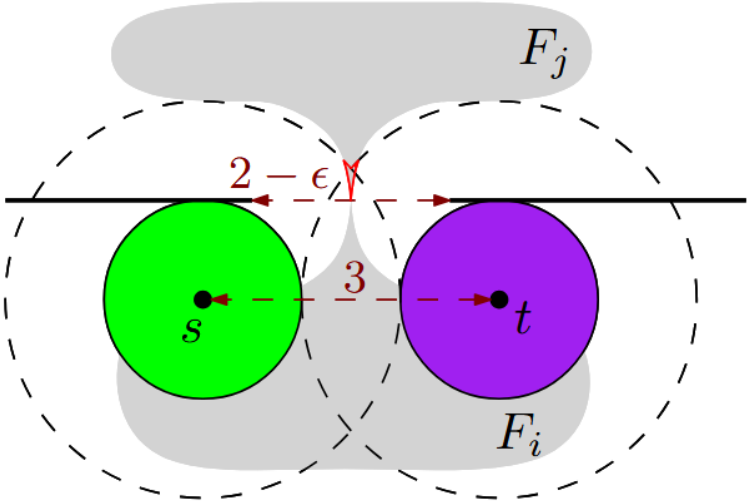
Mehrere Komponenten des Freespace

Interferenzen

Start- und Zielpositionen können mit anderen Freespace-Komponenten interferieren.

Allerdings bleibt die andere Komponente zunächst ohne diesen *blockierenden Bereich* zusammenhängend.

Frage:
Wenn eine Startposition eine andere Komponente durch den blockierenden Bereich in zwei Komponenten teilt, kann das gleichzeitig auch eine Zielposition?

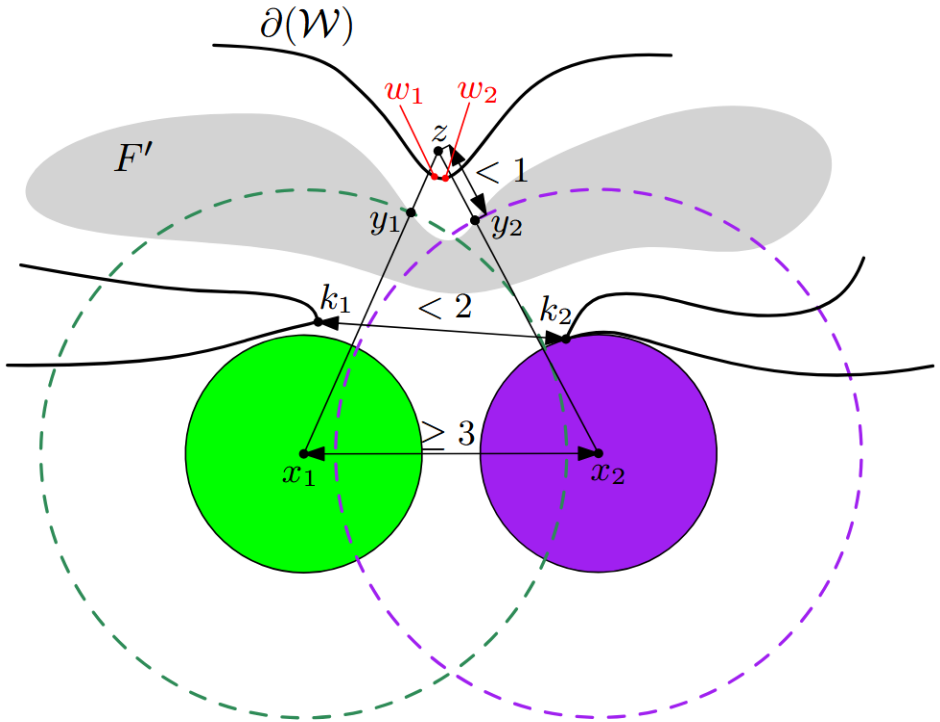


Nur ein *Blocker*

Annahme:
Start und Ziel zerschneiden andere Komponente.

Zeige:
Der Punkt z muss gleichzeitig in und außerhalb des Workspaces liegen.

Damit:
Ordne die Komponenten, in welcher Reihenfolge sie abgearbeitet werden müssen!



Unzusammenhängender Freespace

Theorem:
Gegeben seien m Kreisroboter in einem einfachen Workspace mit Start- bzw Zielpositionen S und T, sowie $\mu \geq 4$ und $\beta \geq 3$.
Besitzt jede Komponente des Freespace die gleiche Anzahl an Start- und Zielpositionen, dann existiert ein kollisionsfreie Bewegungsplan, um die Roboter von S auf T zu bewegen.