1 Problembeschreibung und -analyse

1.1 Problembeschreibung

Die Hauptschwierigkeit des Problems liegt daran, dass es sich eigentlich um eine Kombination aus (mindestens) zwei Problemen handelen. Jedes Problem für sich ist gut beschrieben und gelöst worden, doch für alles gleichzeitig nicht.

1. Graph coloring - Visit stuff
2. Evacuate

1.2 Annahmen über das Problem

- Roboter sind punktförmig
- Können in der Bewegung saugen
- Keine Beschränkung, wie viele Roboter sich in einem Punkt befinden
- Kommunikation ist instantan und ohne Berechnungszeit (Senden wie Empfangen)
- Kommunikation zwischen Robotern kann alles sein
- Roboter haben unbeschränkt viel Speicher

1.3 Disclaimer

Einschränkungen nicht simuliert, aber deren Auswirkungen

2 Vorgehensmodell

- Keep it simple
- Start with working prototype before you do srs stuff
- Getrennt marschieren, zusammen kämpfen (self contained subprograms, assemble at the end, bottom to top)
- ipython notebook
- using a lib before reinventing the wheel
- Optimize when needed, first correct and working, then fast
- 90 % thinking, 10 % coding
- Abstract away decisions not made or which might change (geometry, polygon, agent)

2.1 Design-Ziele

2.2 Nicht-Ziele

Schöne UI, nur usable

3 Problembetrachtung

Das Problem kann unter vielen verschiedenen betrachtet werden. Je nachdem, in welcher Domäne der Informatik es eingeordnet wird, gibt es unterschiedliche Lösungsansätze.

In diesem Abschnitt wird kurz beschrieben,

3.1 Graphenproblem

Suche

3.2 Optimierungsproblem

3.3 Maschinenlernen

3.4 Künstliche Intelligenz

Abbildung 1: Diskretisierung des Spielfeldes mittels verschiedener Polygone

4 Umgebung

Die Modellierung des Meeresbodens hat zum Ziel, die folgenden Fragen zu beantworten oder einen guten Kompromiss zu finden:

1. Wie lassen sich eine begrenzte Anzahl an Kreisen auf einer unendlichen Fläche anorden, sodass die nicht von Kreisen bedeckte Fläche minimal ist (vergleichbar mit dem Ausstechen von Kreisen aus Keksteig)?
2. Wie lässt sich der Meeresboden unterteilen, sodass eine Simulation möglichst einfach wird?

Geplant ist, dass die Missionsdauer in Zeitschritt von 1s aufgeteilt wird. In jedem Zeitschritt bewegen sich die Roboter einen geometrischen Schritt auf dem Meeresboden weiter und säubern ihn dabei von Manganknollen. Der Vorteil darin besteht, dass die Optimierung der Ausbeute darauf reduziert wird, den nächsten Schritt möglichst geeignet auszuwählen.

Dazu werden unendlichen Weiten des pazifischen Meeresgrundes in Abschnitte in Form von Polygonen eingeteilt (parkettiert), damit der Wertebereich der nächsten möglichen Schritte diskretisiert wird.

Die Wahl, mit welchem Polygon gearbeitet wird, hat direkte Auswirkungen auf Genauigkeit und Leistungsfähigkeit der Simulation, wie im Folgenden gezeigt wird.

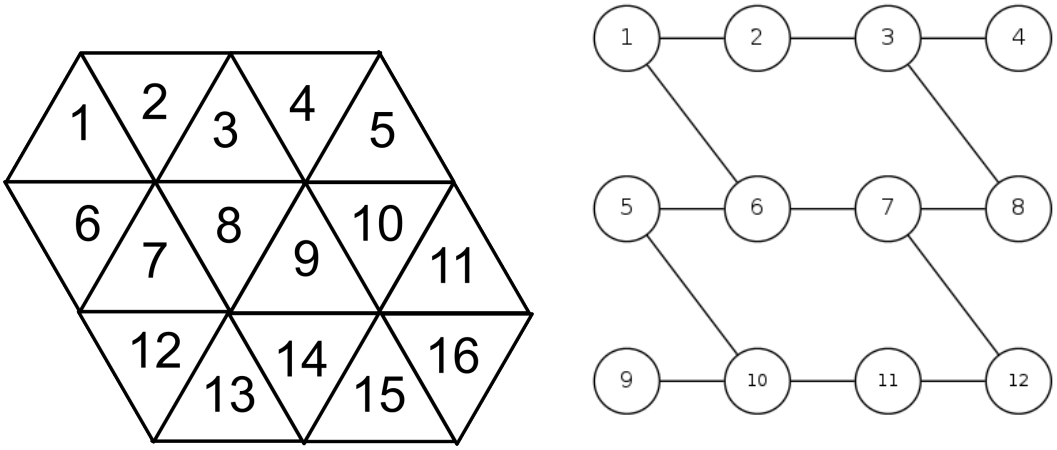
4.1 Konkrete Modellierung des Meeresbodens

Es gibt nur drei verschiedene Polygone, mit denen gleichmäßig ohne Zuhilfenahme von Füllstücken parkettiert werden kann 1 [1].

Der Meeresboden kann dann als Graph gesehen werden, bei denen Knoten als Zellen gesehen werden und geometrisch einem Polygon entsprechen. Angeordnet werden diese so, dass zwischen den Schwerpunkten zweier benachbarter Zellen exakt 1m Abstand ist. Dies hat den Grund, dass ein Roboter in jedem Zeitschritt in die Mitte der nächste Zelle wechseln kann.

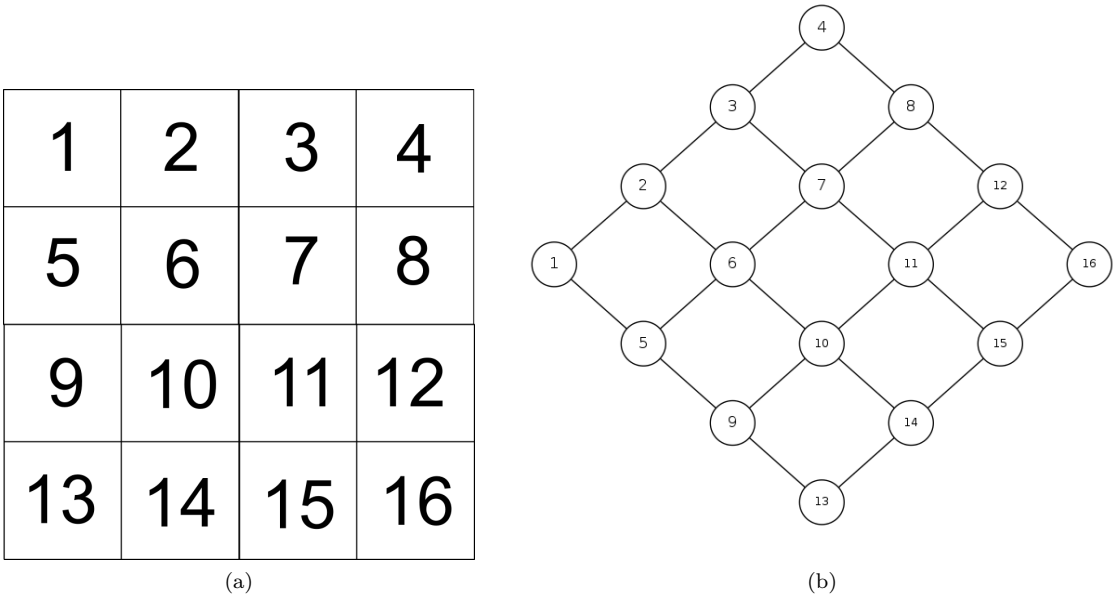
Der Zusammenhang zwischen realer Umgebung und Modellierung kann in Fig.

[1] Source



(a) (b)

Abbildung 2: Parkettierung mit gleichseitigen Dreiecken



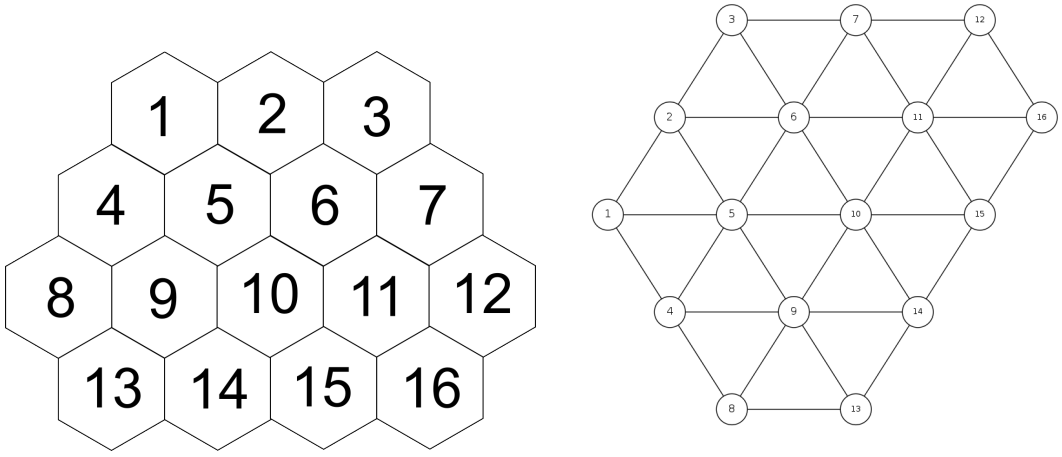
(a) (b)

Abbildung 3: Parkettierung mit Quadraten

Roboter werden so simuliert, dass die Bewegungen nur über die Kanten einer Zelle möglich sind. Daher ist Anzahl der Ecken identisch mit den möglichen Bewegungsrichtungen. Somit gilt: je mehr Ecken, desto mehr Freiheitsgrade. Die Kanten geben an, von welchem Knoten zu welchen Nachbarn gewechselt werden kann. Somit hat jeder Knoten auch so viele Kanten wie das gewählte Polygon Ecken hat.

Da eine Simulation mit unterschiedlichen Polygonen unnötig komplex wird, entscheidet es sich zwischen gleichschenkligem Dreieck, Quadrat und regelmäßigem Sechseck.

Der Arbeitsbereich eines Roboters ist kreisförmig, die Zellen, in denen er sich befindet, jedoch ein Polygon. Daher gibt es in den Ecken der Zelle Bereiche, die nicht (unmittelbar) gesaugt werden.



(a) (b)

Abbildung 4: Parkettierung mit regelmäßigen Sechsecken

Berechnen lässt sich der Verlust über das Verhältnis von der Fläche des Innenkreises des Polygons zum Flächeninhalt des Polygons selbst.

Die folgende Tabelle zeigt, wie viel Prozent einer Zelle nicht gesaugt werden, je nachdem, welches Polygon gewählt wurde.

Die Auswahl fiel schließlich leicht, da die Simulation mit einem Quadrat als Polygon folgende Vorteile bietet:

- Einfache Datenstruktur
- Einfaches Berechnen der Nachbarn
- Visualisierung entspricht pixeln, keine Umrechnung nötig
- Weniger Verlust als Dreieck
- Weniger Freiheitsgrade als Sechseck (weniger Auswahl bedeutet weniger Rechenaufwand)
- Einfache Berechnung von Entfernungen zwischen zwei Zellen 5.1

Die Herleitung kann im Anhang [2] nachvollzogen werden.

[2] Anhang

Abbildung 5: Innenkreis und Polygon

Abbildung 6: Fehlerrechnung

5 Theory Crafting

5.1 Manhattan-Metrik

5.1.1 Entfernung

5.1.2 Kreise

5.2 Platzierung

Keine Häufigkeitsverteilung gegeben

5.3 Missionsdauer

goal runden missiontime runden

5.4 Finden des Sammelpunktes

5.5 Zeitbeschränkung

6 Implementierung

7 Diskussion

7.1 Performance

7.2 Fehlerrechnung

7.2.1 Abweichung vom Optimum

7.2.2 Fehler durch Modellierung

7.3 Lessons learned/Fehlentscheidungen

- Zu viele Abstraktion, die sich später unnötig herausgetsellt hat (Geometry, Circle)

7.4 Ausblick