Projekt - Nachtrag

Vorüberlegung

Graph

Ein (ungerichteter) **Graph** $G = (V, E)$ besteht aus

- einer Menge V von Knoten (*vertices*) und
- einer Menge ungeordnete Knotenpaare $E \subseteq V \times V$ von Kanten (*edges*).

Zu jedem Zeitpunkt kann für jeden *agent* aus der aktuelle Spielsituation ein ungerichteter Graph erzeugt werden. Für den *agent* α wird der Graph G_α erzeugt, wobei β der andere *agent* ist.

- Für jede *site* wird ein Knoten erzeugt, wenn sie nicht von β blockiert ist.
- Für jeden vorhandenen *link* wird eine Kante erzeugt, wenn keine der *sites*, die er verbindet, von β blockiert ist.

Zusammenhangskomponente

Für eine **Zusammenhangskomponente** $Z \subseteq V$ eines ungerichten Graphen $G = (V, E)$ gilt Folgendes.

- Für alle Knotenpaare $v, w \in Z$ gibt es einen Folge von Kanten $e_1, \ldots, e_k \in E$ mit
$$e_1 = \{v, x_1\} \qquad e_i = \{x_{i-1}, x_i\} \text{ für } i = 2, \ldots, k-1 \qquad e_k = \{x_{k-1}, w\}\ .$$
- Für alle Knoten $v \in V \backslash Z$ gilt, es gibt keine Kante $\{v, w\} \in E$ mit $w \in Z$.

D.h. eine Zusammenhangskomponente umfasst alle Knoten, zwischen denen es eine Weg gibt.
Hinweis. Ein Graph kann mehrere Zusammenhangskomponenten haben.

Tiefensuche

Die Zusammenhangskomponenten eines Graphen $G = (V, E)$ können mit **Tiefensuche** (*depth-first search*) bestimmt werden.
Der folgende rekursive Algorithmus `DFS`, der eine Referenz auf einen Knoten $v \in V$ und eine Referenz auf eine Menge von Knoten $Z \subseteq V$ übergeben bekommt, fügt zu Z alle Knoten aus der Zusammenhangskomponente hinzu, in der sich v befindet. D.h. um mit `DFS` die Zusammenhangskomponente Z zu bestimmt, in der sich v befindet, initialisiert man Z mit $\emptyset$ und führe `DFS`(v, Z) aus.

`DFS`(v, Z)

1. wenn $v \in Z$, überspringe 2. und 3.
2. $Z = Z \cup \{v\}$
3. für alle $w \in V$ mit $\{v, w\} \in E$ führe `DSF`(w, Z) aus

Gewicht

Das **Gewicht** einer Zusammenhangskomponenten $|Z|$ ist Summe aus Anzahl der Knoten und Anzahl der Kanten, die zur Zusammenhangskomponente Z gehören.

Einfache Strategie

Spielphase 1

Für diese Phase gibt es keine Strategie, die Links werden wahllos entfernt.

Spielphase 2 und 3

Vorrangig gelten die beiden folgenden Regeln.

1. Gibt es einen Zug mit dem das Spiel gewonnen werden kann, dann wird dieser ausgeführt.
2. Ein Zug, der zur Folge hätte, dass der Gegner im nächsten Zug gewinnen könnte, wird nur ausgeführt, wenn es keine Alternative gibt.

Bewegungsspielraum

Der eigene *agent* sollte so viel Bewegungsspielraum wie möglich, der gegnerische so wenig wie möglich haben.

Für die aktuellen Spielsituation werden die Graphen $G_r = (V_r, E_r)$ für den roten und $G_b = (V_b, E_b)$ für den blauen *agent* erzeugt. Aus den Graphen wird für den roten *agent* die Zusammenhangskomponenten Z_r, in der er sich befindet, und für den blauen die Zusammenhangskomponenten Z_b, in der dieser sich befindet, bestimmt.

Dann wird diese Spielsituation für Rot mit $\sigma_r = |Z_r| - |Z_b|$ und für Blau mit $\sigma_b = -\sigma_r$ bewertet.

Vorausschauen

Bei der Auswahl des nächsten Zuges ist es sinnvoll, nicht die Situation nach diesem Zug zu bewerten, sondern erst die Situation, die sich aus der Reaktion des Gegners auf diesen Zug ergibt.

Wenn Blau am Zug ist wird wie folgt verfahren.

1. Für einen Zug von Blau wird jeder mögliche Zug von Rot betrachtet, beide Züge werden ausgeführt und die so entstandene Spielsituation wird aus der Sicht von Blau mit σ_b bewertet. Das Minimum dieser Bewertungen ergibt die Bewertung für diesen Zug von Blau.

2. Nachdem in 1. alle möglichen Züge von Blau bewertet wurden, wird der nächste Zug von Blau zufällig aus den Zügen mit der besten (maximalen) Bewertung ausgewählt.

Für die Auswahl des nächsten Zugs von Rot wird analog verfahren.

<u>Hinweis</u>

Wenn man die Bewertung noch um die Gewinnsituation ergänzt, entspricht dieses Vorgehen dem Minimax-Algorithmus mit Suchtiefe zwei (Halbzüge).