戦略調査\_三浦

実装のポイントまとめ

引用：たぶん、決勝の一回戦で敗退した人のブログ

<http://kumikomiya.com/samurai-jockeys-algorithm/>

※は三浦の意見

■衝突判定

自分の指そうとする手が壁に阻まれるのかどうか判断すること

-実装の手間がかかる

-判定をミスると完走ができない

-実装のポイント

「一つポイントを挙げるとすると、判定結果をキャッシュしておくというのは高速化のために必須です。たとえば、ともに視界の範囲にある点Ｐ－Ｑ間の壁の有無が一度判明したなら、その事実は未来永劫変わりません。そこで、1度目の結果を unordered\_map に格納しておいて、2度目以降はそこから取り出すのです。」

※キャッシュがよくわからないんでもし実装するなら教えて下さい...

■相手の動きの考慮

相手の動きを考慮に入れて、全ての曲面を計算しようとするとC++でも数分かかる。持ち時間の範囲で読めるのはせいぜい4手先。

→枝刈りなど効率よい方法が必要

■めりこみ問題

例えば、壁から抜け出すのに最短で5手必要な場合を想定。4手先しか読めないプログラムでは、脱出ルートを発見できない。永遠に壁にハマり続ける。

-場当たり的な対処:

→壁に当たったときは速度落とすなど

-先読みで対処

→枝刈りなどで純粋に計算コストを小さく、より先まで読めるプログラムを目指す。

■枝刈り

全てのパターンを考慮に入れると制限時間におさまらない。

明らかに最適ではない手は削りたい。

※評価関数を決めたら色々試したいっすね！！

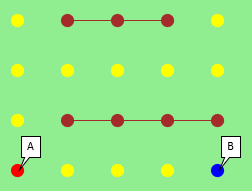
■評価関数

ゲームの目標：y座標を大きくする

→コースによっては値が同じでも有利不利が発生。そのままy座標の大きさを評価関数に

するのは微妙。

例)下記ではAが優勢。



-最短経路探索のアルゴリズムを使う方法。ここではダイクストラ法を使ってる。

※簡単なアルゴリズムを複数試して比較していくといいのかな

-距離が同じなら、選択肢が多いルートのほうが良いことを評価関数に盛り込む。

【感想】

この人は機械学習を使っていないけど、最適ではない手を削る時に機械学習が上手く使えんかな？と思う。最初は何個か有名なアルゴリズムを試していいやつを使う方法がいいかな？

ググったら何個か経験者の記事とかブログがあったので、行き詰ったらそこらへんうまく活用しながらやっていきたいね。