DDR 足運びコスト最小化問題

maton

0.1. マルコフ決定過程としての足運び

0.1.1. マルコフ決定過程と足運びとの対応付け

有限マルコフ決定過程 (finite Markov decision process; finite MDP) は、次の 4 要素 $\langle S, A, T, R \rangle$ を持つ。

- $S = \{s^1, s^2, ..., s^N\}$: 状態の有限集合
- $A = \{a^1, a^2, ..., a^K\}$: 行動の有限集合
- $T : S \times A \times S \to [0, 1]$: 遷移関数
- $R : S \times A \times S \to \mathbb{R}$: 報酬関数

DDR における足運びはあくまで時空間的に連続的な運動であるが、譜面のあるノーツを踏むという点で区切ることで、時間的に離散的に考えることが可能になる。また、パネルのある 1 点を踏むと簡略化して考えることで、空間的にも離散的に扱うことができる。足運びの基本単位を足配置とし、あるノーツを正確な位置で、正確なタイミングで踏むことを想定する。このとき、足配置は MDP における状態 $s \in S$、ある足配置から次の足配置に向けた移動を行動 $a \in A$ とみなすことができる。MDP における遷移関数 T は次状態への確率的遷移を表すが、行動 a が足配置間の決定的な移動を表すため、あらゆる $(s, a, s') \in S \times A \times S$ において、遷移確率は 1 となる。報酬関数 R は、最大化問題において、良いアクションに対して高い報酬を与える関数となるため、コスト最小化問題においてはコストをより低くするようなアクションに対して高い報酬を与える関数として定義する。また、アクションによって所望のノーツが踏めていない場合の報酬はゼロになる。

0.1.2. 各要素が持っておくべき情報

具体的な計算を行うためには、MDP の 4 要素がどのような情報を持っておくべきかを定めなければならない。

状態 s は、現在の時刻と足配置を持つ。足配置は足と座標の組の列である。これは、プレイヤーの時間的、空間的な位置を表している。簡単な実装の例では、時刻はノーツと同期する離

散時間として、足の座標はパネルを指定するものとする。ノーツ間にアクションが必要なら時間を更に細かく刻むことも考えられる。また、パネル外の配置が必要になったり、パネルの中でも踏む位置を詳細に指定する必要があれば、座標を更に細かく刻むことも考えられる。

行動 a は、足配置を移動させる行動である。片足だけのアクションならば、交互やスライドを選択することになる。一方、両足を用いるアクションでは、足の対応関係、さらには捻る方向まで考慮すべきである。

遷移関数 T は、アクションに対して想定通りの足配置を取ることができる度合いを計算する。理想的な足運びを出力するという目的においては、遷移確率は常に 1 とする。逆に、プレイヤーの能力に一定の制約を科す場合は、コストの大きい遷移の確率を下げることなどが考えられる。

報酬関数 R は、ある時刻にある足配置を取った際の報酬を計算する。ノーツ情報はここで使用することになる。ある時刻に降ってくるノーツを正しく踏めていれば、コストの逆数を報酬とする。ノーツを正しく踏めていなければ、ゼロに近い（またはマイナス）報酬を与える。

0.2. コストの考え方

0.2.1. 重心を用いたコストの考え方コストの構成要素

DDR におけるコストは、図1で示すように次の 5 種類に大別することができる。

- Interval Cost: ノーツの間隔から算出されるコスト。短ければ短いほどコストは高く、一方で、ある一定以上の間隔が空いていればコストは下限値を取ると考えられる。
- Foot Cost: 足の選択から算出されるコスト。交互、スライド、同時といった選択肢によってコストが変化する。基本的に交互で踏むとコストは低くなるが、ノーツ列によってはスライドを入れることで総コストを下げられるケースがある。
- Move Cost: 足の移動距離から算出されるコスト。長ければ長いほどコストが高い。特殊なケースとして、全く移動しない場合がある。
- Direction Cost: 体の向きから算出されるコスト。捻りの具合が大きいほど踏みにくくなる。正面を基準に、左右方向への振れ幅が大きければ大きいほどコストが高い。
- Angle Cost: 体の向きの回転角から算出されるコスト。捻る角度が大きいほど踏みにくくなる。元の位置を基準に、左右方向への振れ幅が大きければ大きいほどコストが高い。

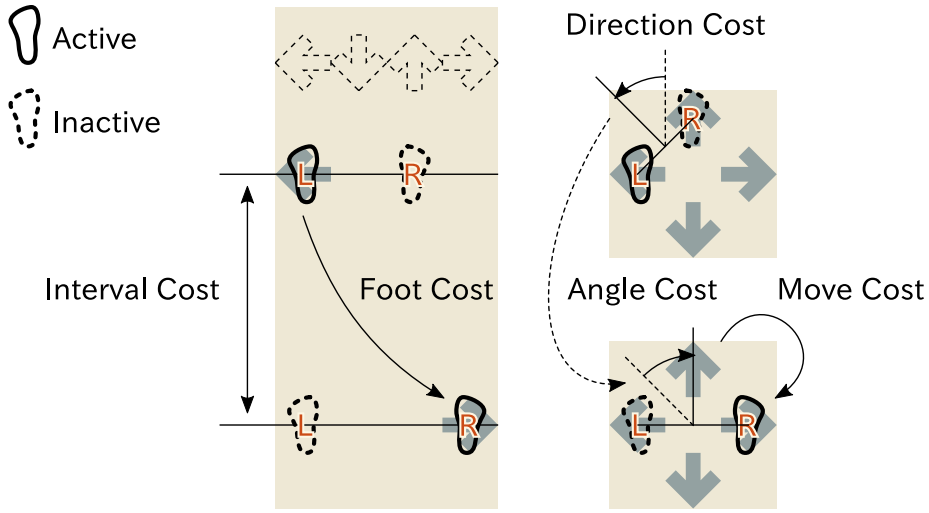

図 1 コストの構成要素

本項では、どちらの足で踏むか、どのように体を動かして踏むか、という方策に対して、上記のようなコストを既知のものとして与えることで、コストを最小化する方策がプレイヤーの自然な意思決定と等価となることを目指す。

一方で、この問題を繊細に扱うには、古典力学を用いてエネルギー消費を最小化する問題を解くことも有効だと考えられる。本稿時点ではそのアプローチを直接扱わず、今後の課題とする。

0.2.2. 重心を用いたコストの考え方

図1で導入したコストを個別に算出して合算することも考えられるが、ここでは、左右の足の重心の移動距離を用いてコストを考える。図2の上図は、右足を → パネルに置いている状態で ← パネルを左足で踏み、次の ↑ パネルを交互に踏むか、スライドするかの選択肢を表している。

図2の左図は、左右の足の間の位置を重心として、重心の移動距離をコストとみなすアイディアであり、Move Cost を表現することができる。この方法では、Interval Cost はもちろん、重心を通る線に対する垂線を体の向きと考えることができるため、Direction Cost や Angle Cost を扱うことができる。一方、この方法では交互の場合とスライドの場合でコストは等しいと見なされ、Foot Cost に対する直感は反映されていない。

そこで、図2の右図のように、重心の位置をパネルを踏んだ側にずらす方法を導入する。こ

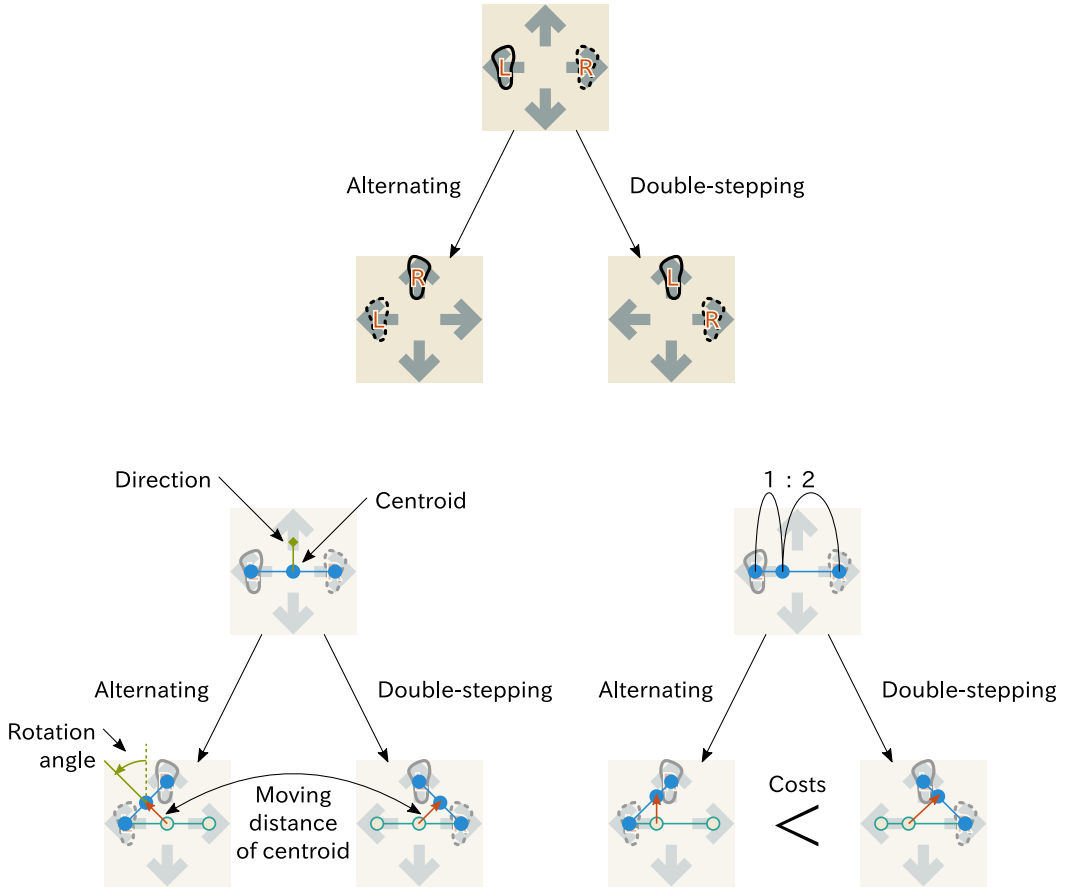

図 2 重心を用いたコストの考え方

の方法では、踏んだパネルの中心と足を置いているパネルの中心を 1 : 2 で分割する位置を重心とする。この方法により、交互で踏んだ場合に比べ、スライドした場合のほうが重心の移動距離が大きくなり、Foot Cost に対する直感に近い表現となる。

0.2.3. 捨て譜面の考え方

時にプレイヤーは、譜面の一部を無視することで高いスコアを狙う戦略を取る。譜面の一部を無視するすることで得られる譜面を捨て譜面という。原則として、満点を取る場合はすべてのノーツを拾う必要がある。しかし、非常にコストが高いノーツ系列に対して、プレイヤーのスキルが不足している状態では満足の得られるスコアを出すことができない場合がある。そのため、譜面の一部を捨て、プレイヤーの現在のスキルで対処可能なレベルに簡易化することで、元の譜面より高いスコアを狙うという戦略が、攻略上採用されることがある。

捨て譜面の戦略を取り入れるには，足運びのコストだけを考慮する報酬関数では不十分で

ある。捨て譜面によるコストの削減と、得られるスコアの期待値減少のトレードオフを考慮しなければならない。そのため、報酬関数は「コストの少なさ」だけでなく、「スコアの期待値の高さ」も構成要素となる。

また、捨て譜面によってダンスゲージが減少することを考慮しなければならない。ダンスゲージがゼロになるとクリア失敗となるため、譜面の捨てすぎも避けなければならない。

一方、どの程度譜面を捨てなければならないか、その譜面でダンスゲージがゼロにならないか、といった問題を解くことにより、「現在のプレイスキルである譜面がクリアできるか」という問題が解ける可能性がある。この場合、コストやスコア期待値の算出において、プレイヤーのスキルを考慮する必要があるだろう。

0.2.4. 空打ちの考え方

時にプレイヤーは、ノーツが存在しないタイミングで特定の位置を踏み、次のノーツに備える空打ちという戦略を取る。空打ちが果たす役割は主に次の 2 つになる。

- リズムを維持する。次のノーツがまで長い時間が空いている場合や停止後に突然動き出す譜面に対してタイミングを計ったり、裏拍のリズムを正確に踏むために表拍のリズムを刻んでタイミングを計ったりする。
- 姿勢を維持する。続けて踏むにはスライドが必要となる場面で、空打ちをすることで交互に踏めるようにしたり、長距離の移動が必要な場面で、間に空打ちを挟むことで姿勢を崩さずに移動したりする。

空打ちの戦略を取り入れるには，ノーツごとにアクションを行うようなモデリングでは不十分である。ノーツとノーツの間で必要に応じて適宜アクションを実施できるようなモデルでなければならない。例えば、ノーツ単位の離散時間の代わりに、もっと細やかな（例えば数ミリ秒単位の）離散時間または連続時間を採用することなどが挙げられる。