# 1 E.A.3.3 (ProteinFolding)

## 1.1 Modello

Una proteina è una sequenza di amminoacidi $a_1, ..., a_n \ \ t.c. \ \ a_i \in \{H, P\}$.

**Def. 1 conformazione (stato)**

Una conformazione è un assegnamento $p_1, .., p_k$ con $0 \leq k \leq n \ \ t.c.$

- $p_i \in \mathbb{Z}^2$ *(posizione)*
- $\text{dist}(p_i, p_{i+1}) = 1$ *(adiacenza)*
- $\nexists p_i, p_j \ \ i \neq j \wedge p_i = p_j$ *(non sovrapposizione)*

**Def. 2 conformazione completa (obiettivo)**

Uno stato è obiettivo quando $k = n$.

**Def. 3 azione**

Dato uno stato **non obiettivo** (quindi $k < n$), l'insieme di azioni consiste nei possibili assegnamenti per $p_{k+1}$ che rispettano le condizioni di *adiacenza* e *sovrapposizione*. Un'insieme di azioni $A = \{\alpha \in \mathbb{Z}^2\}$.

Da questa modellazione del problema deriva un'importante osservazione, che permette di ottimizzare gli algoritmi di ricerca:

**Oss. 1**

Il grafo di ricerca per **ProteinFolding** è un **albero**.

**Dim.** Per dimostrarlo bisogna far vedere che non è possibile raggiungere lo stesso stato con sequenze di azioni diverse. Per assurdo, se fosse possibile, vorrebbe dire che uno stato $p_1, ..., p_k$ è stato raggiunto partendo da due stati $q_1, ..., q_{k-1}$ e $r_1, ..., r_{k-1}$, in uno di due modi:

1. $q_i = r_i$ per ogni $0 \leq i \leq k-1$, e $q_k \neq r_k$, quindi, per come sono definite le azioni, $p_k = q_k \neq r_k = p_k \Rightarrow p_k \neq p_k \rightarrow\leftarrow$
2. $\exists q_i \neq r_i$ per un qualche $0 \leq i \leq k-1 \Longrightarrow p_i = q_i \neq r_i = p_i$, ma questo non è possibile, perché non ci sono azioni che riassegnano una posizione già assegnata $\rightarrow\leftarrow$

L'**Oss. 1** permette di ottimizzare la ricerca, perché non serve controllare se uno stato è già stato esplorato o sta già in frontiera (ogni stato generato è nuovo, quindi non sta né in frontiera né nell'isieme degli esplorati).

Dato che il problema presenta forti **simmetrie**, si può ridurre lo spazio di ricerca evitando le azioni che generano stati simmetrici o ruotati, permettendo solo le seguenti azioni:

- $p_0 = (0, 0)$

- $p_1 = (0, 1)$
- per $p_i$ con $i > 1$
  1. se $\exists y_j \forall j \;\; j \leq i \wedge p_j = \big(0, y_j\big)$, dati $p_{i-1} = (0, y_{i-1})$ le azioni possibili sono $(0, y_{i-1} + 1)$ (andare avanti) e $(1, y_{i-1})$ girare a destra (queste due sono sempre possibili per come è costruita la catena, TODO: ottimizzare questo caso per non controllare se sono valide, altrimenti andare avanti, metterlo sotto forma di osservazione / teorema)
  2. altrimenti, $A = \{\}$

**Oss. 2**

Le azioni nel caso 1. rispettano sempre la non sovrapposizione (rispettano la posizione e sono adiacenti per costruzione)

- l'azione $(0, y_{i-1} + 1)$:
  - possibile perché se ci fosse un nodo in quella posizione questo avrebbe distanza euclidea 2, in contrasto con l'adiacenza (magari lo dovrei fare induttivamente?)
- l'azione $(1, y_{i-1})$ è sempre possibile perché $\forall y_j, j \leq i - 1$ si ha $x_j = 0$

**Oss. 3**

Usando le regole sopra non si generano stati simmetrici, ruotati o traslati.

**Dim**. Per induzione su $k$

- $k = 0$: …
- $k = 1$: …
- $k = 2$: …

**Passo induttivo**

- se c'è un giro
  - allora la proteina è una retta, dato che non ci sono cose simmetriche sotto, non può generare roba simmetrica andando avanti, e facendo il giro a destra non c'è roba ruotata
- se non c'è un giro
  - eh, bella per me, ma dato che non sono simmetriche quelle sotto non è simmetrica neanche quella sopra, come riottengo l'ipotesi induttiva $k$ per una proteina lunga $k + 1$

## 1.2 Euristica

Due parole veloci sul come l'idea di fondo è che proteine H vicine nella sequenza devono stare vicine nella proteina finale, o, in generale gli stati più «promettenti» sono quelli che hanno «più contatti» per ora.

TODO:

- euristica 2 (e calcolo dell'euristica)

**Oss. 4 Ammissibilità**

**Oss. 5 Consistenza**

**Oss. 6 Critical ratio**

## 1.3 Possibili miglioramenti

- levare l'`Rc<AminoAcid>` (il reference counter), ma servirebbe modificare l'interfaccia per il `Problem`. L'obiettivo sarebbe possibilmente usare un `usize` per riferirsi allo stato.

```
pub struct AminoAcid {
    pos: Pos,
    prev: Option<Rc<AminoAcid>>,
    depth: usize,
    first_turn: bool,
}
```

- verificare in modo più intelligente l'ammissibilità di un'azione senza dover scorrere tutta la proteina
- alternativamente, trovare una conformazione in memoria per gli stati in modo da tenere vicini gli stati visti più di frequente, e quelli «inutili» lasciarli in fondo