Friday, June 2, 2023 10:02 AM

(1)

2891 = 1 0 1 1 0 1 0 0 1 0 1 1

3927 = 1 1 1 1 0 1 0 1 0 1 1 1

1 1 1 1 ⟹ Hamming Distanz = 4

(2)

Der Genotyp hat die Länge $n=46$, also 45 mögliche Crossover spaltungen. Angenommen ein Crossover findet zu 100% statt, ergeben sich folgende Stellen inmitten der funktional zusammengehörender Gruppe:

101001**0**101000101101101**01**01010010101**1**00**101**110111

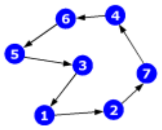
1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 10 1 2 3 4 5 6 7 8 9 10 1 2

32

Das bedeutet, wenn an 32 der 45 möglichen crossover stellen gespalten wird, ist die Gruppe aufgebrochen.

$\frac{32}{45} = 0{,}7\overline{11} = 71{,}\overline{11}\,\%$

(3) Pfadcodierung P = (1 2 7 4 6 5 3)



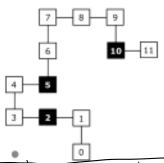
Kantencodierung K = (2 7 1 6 3 5 4)

Originalcodierung O = (1 1 5 2 3 2 1)

Stadt	Liste	Pos/Codierung
1	1→2→3→4→5→6→7	1
2	2→3→4→5→6→7	1
7	3→4→5→6→7	5
4	3→4→5→6	2
6	3→5→6	3
5	3→5	2
3	3	1

(4) 1. Absolute Gitterkoordinaten

P_0 (3/1)	P_3 (1,2)	P_6 (2,4)	P_9 (4,5)
P_1 (3,2)	P_4 (1,3)	P_7 (2,5)	P_{10} (4,4)
P_2 (2,2)	P_5 (2,3)	P_8 (3,5)	P_{11} (5,4)



P_0 (0/0)	$P_{2\to3}$ (0/-1)	$P_{5\to6}$ (1/0)	$P_{8\to9}$ (0/1)
$P_{0\to1}$ (1/0)	$P_{3\to4}$ (1/0)	$P_{6\to7}$ (1/0)	$P_{9\to10}$ (-1/0)
$P_{1\to2}$ (0/-1)	$P_{4\to5}$ (0/1)	$P_{7\to8}$ (0/1)	$P_{10\to11}$ (0/1)

2. Relative Gitterkoordinaten

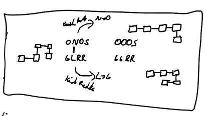
P_0 (0/0)	$P_{2\to3}$ (L)	$P_{5\to6}$ (O)	$P_{8\to9}$ (R)
$P_{0\to1}$ (O)	$P_{3\to4}$ (O)	$P_{6\to7}$ (O)	$P_{9\to10}$ (U)
$P_{1\to2}$ (L)	$P_{4\to5}$ (R)	$P_{7\to8}$ (R)	$P_{10\to11}$ (R)

3. Relative Konformationskoordinaten

P_0 (0/0)	$P_{2\to3}$ (G)	$P_{5\to6}$ (LG)	$P_{8\to9}$ (G)
$P_{0\to1}$ (G)	$P_{3\to4}$ (RG)	$P_{6\to7}$ (G)	$P_{9\to10}$ (RG)
$P_{1\to2}$ (LG)	$P_{4\to5}$ (RG)	$P_{7\to8}$ (RG)	$P_{10\to11}$ (LG)

Epitasie messen:

1. Wenn in absoluten Koordinaten ein Punkt verschoben wird, verändern sich alle anderen ohne weiteres erstmal gar nicht!
   ⇒ Es entsteht eine Lücke in der Proteinfaltung und eine Epitasie im Phänotyp ab der Position, an der die Verschiebung stattfand.
   Das kann alle Positionen $n-2$ bis $n-(n-1)$ betreffen.
   Epitasie = Abhängigkeit (Anzahl aller Gene nach verschobenem.
2. In relativen Gitterkoordinaten kann bei einer Mutation keine Lücke mehr entstehen, da Folgekoordinaten anhand vorheriger und einer Himmelsrichtung errechnet werden.
   Jedoch besteht die Möglichkeit, dass ab dem Mutationspunkt, durch die Folgekoordinaten mit fester Richtung, die sich nicht auto. ändert, eine fälschlich andere Faltung entsteht und somit doch Epitasie entsteht. Keine funktionale "Ablopplung" aber anderes Layout.

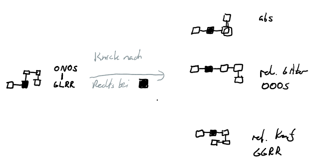


3. Außer auftretender Überlappungen, wie auch bei 1 & 2 möglich. Behält die Folgefaltung ab dem Mutationspunkt ihre Struktur und hat somit die geringsten Epitasiechancen, bzw. gar keine Änderung im Phänotypabschnitt nach dem Mutationspunkt

---

Geringste Epitase

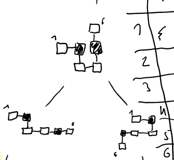
Die relativen Konformationskoordinaten!



Algorithmus für Überlappung bei relativen Konformationen ohne Koordinaten!

Actual Lösung aus P

Per Abstandsmatrix



	1	2	3	4	5	6
1	ε	1	$\sqrt{2}$	$\sqrt{5}$	2	$\sqrt{5}$
2		ε	1	$\sqrt{2}$	1	$\sqrt{2}$
3			ε	1	$\sqrt{2}$	$\sqrt{5}$
4				ε	1	$\sqrt{2}$
5					ε	1
6						ε

d=7!

	1	2	3	4	5	6
1	ε	1	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{2}$
2	/	ε	1	$\sqrt{2}$	$\sqrt{5}$	$\sqrt{8}$
3	/	/	ε	1	2	$\sqrt{5}$
4	/	/	/	ε	1	$\sqrt{2}$
5	/	/	/	/	ε	1
6	/	/	/	/	/	ε

d=5

5 Unterschied

	x	y
P_2	1	0
P_5	-1	-1

Unterschied: 2 1

$\sqrt{2^2+1^2} = \sqrt{5}$

Algorithmus
- Liste aller Koordinaten erstellen
- Distanzvergleich zu nächsten Punkten:

function checkOverlaps(List<Direction> d) // ... Stück to check overlaps ...

(function checkOverlapsByCoords (List<Direction> d)

d = 0 → overlap

ref. Konf
GGRR

Algorithmus für Überlappung bei relativen Konformationen ohne Koordinaten!

```
function checkOverlap(List<Directions> d)      // use Stack to check overlaps for rel bitter.
{                                              //  if we move North 2 times and South 2 times same pos.
  Array<Directions> a = []
  int overlaps = 0
  bool started = false;

  for(int i=0; i<d.Count(); i++)
  {
    if (a.Count == 0)
      a.push(d[i]);
      if (started)   // we started and found an overlap due to opposing directions
        overlaps++;
    else
    {   // if we move in opposing directions remove -> same place
      if (a[a.Last()] == NORTH && d[i] == SOUTH
        || a[a.Last()] == EAST && d[i] == WEST
        || a[a.Last()] == SOUTH && d[i] == NORTH
        || a[a.Last()] == WEST && d[i] == EAST)
          a.remove(a.count());

      else
        a.push(d[i]);   // other direction: push
    }
    started = true;
  }
  return overlaps;
}
```

Algorithmus
- Liste aller Koordinaten erstellen
- Distanzvergleich zu nächsten Punkten:
  - d = 0 → overlap
  - d > j → break, weil für diesen Punkt schaffen wir in j Schritten nicht mehr zum Punkt zurück zu kehren!
  - j = i-2 → vorbei, die beiden Vorletzten können nicht mehr overl.

```
function checkOverlapsByCoords (List<Direction> d)
{
  List<Point> visited = [];
  int overlaps = 0;
  currPoint = (0,0)

  for(int i=0; i<d.Count(), i++)
  {
    if (currPoint in visited)
      overlaps++;
    else
      visited.add(currPoint);

    if d.at(i) == Direction:North
      currPoint = (currPoint[0], currPoint[1]+1);
    if d.at(i) == Direction: East
      currPoint = (currPoint[0]+1, currPoint[1]);
    if d.at(i) == Direction: South
      currPoint = (currPoint[0], currPoint[1]-1);
    if d.at(i) == Direction: West
      currPoint = (currPoint[0]-1, currPoint[1]);
  }
  return overlaps;
}
```

Coding ohne overlaps

Koordinatensystem Klasse mit Feldern und bool besetzt.
Viel Speicherverbrauch!

Class Tabelle   besetzt = true.

	7	8	9	
	6		10	11
4	5			
3	2	1		
		0		

→ aber absolute Koordinaten
– wo gehört eine Mutation hin?

Aufgaben in der Klausur kommen btw kaum 3* dran ;-)