

1. To find zero-force target location (\hat{x}_i, \hat{y}_i)

$$\sum_j w_{ij} \cdot (\hat{x}_j - \hat{x}_i) = 0 \Rightarrow \hat{x}_i = \frac{\sum_j w_{ij} \hat{x}_j}{\sum_j w_{ij}} \Rightarrow \frac{2 \times 1 + 5 \times 4 + 3 \times 2 + 8 \times 6}{18} = 4.22 = \hat{x}_i$$

$$\sum_j w_{ij} \cdot (\hat{y}_j - \hat{y}_i) = 0 \Rightarrow \hat{y}_i = \frac{\sum_j w_{ij} \hat{y}_j}{\sum_j w_{ij}} \Rightarrow \frac{2 \times 1 + 5 \times 3 + 3 \times 5 + 8 \times 2}{18} = 2.66 = \hat{y}_i$$

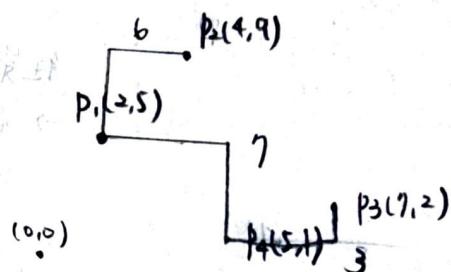
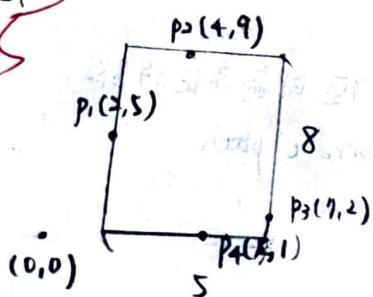
若只能放在整数位置

$$\rightarrow \text{四舍五入} \rightarrow (4, 3) = (\hat{x}_i, \hat{y}_i)$$

又 (4, 3) 已被佔領

\rightarrow 可以選相鄰的點 (4, 2)

15



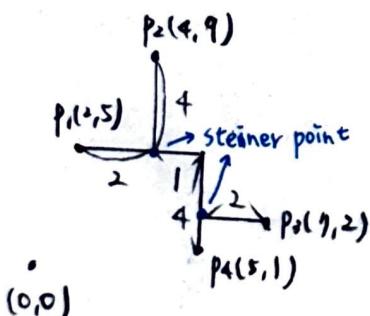
half-parameter approximation

找最小能全部框起來的矩形的

半周長 $P = 5+8=13$

minimum cost spanning tree

$$f = 6+7+3 = 16$$



minimum cost Steiner tree

$$r = 4+2+1+4+2 = 13$$

3. Quadratic wirelength

pros: convex, smooth function \rightarrow 求極值可進行微分 \rightarrow easy to minimize

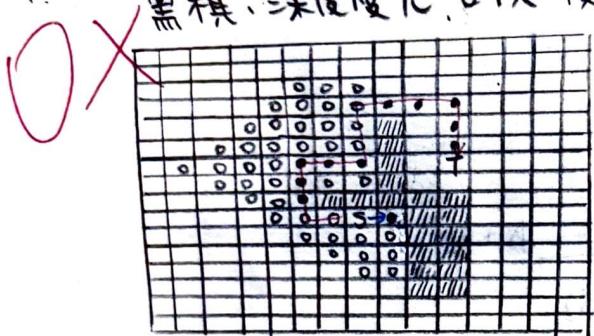
cons: 算出來的值非實際 wirelength, 還需另外處理

non-quadratic

pros: 算出來的值為實際的 wirelength, 可直接用來估計

cons: not differentiable \rightarrow 較難求極值

4. 黑棋: 深度優先, 白棋: 寬度優先 四: 障礙走物



往下方向有路就做 Depth-first Search
沒路做 Breadth-first Search 直到

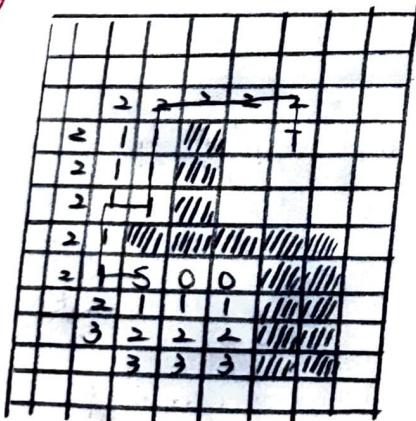
跨過障礙走物

往第一顆黑棋走的路徑即為多走的路
 \rightarrow 有可能不是找到 shortest path

shortest path.

5.

Hadlock's algorithm



從 label i

如果往下方向走 label = i+1

反方向 label = i

not clear

No.

圖為 slicing tree \rightarrow 先走的 channel 要後繞.

slicing tree 的順序為 $c \rightarrow a \rightarrow b$

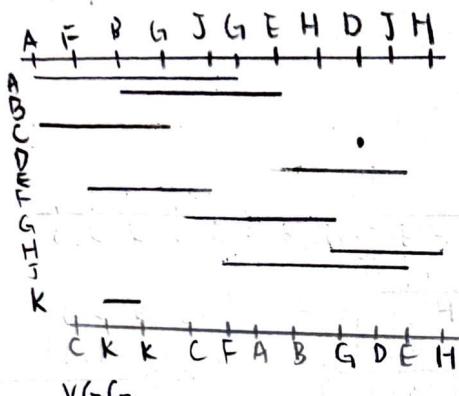
因此繞的順序應為 $b \rightarrow a \rightarrow c$

否則 a 先绕完可能就會讓 b 繞不出來.

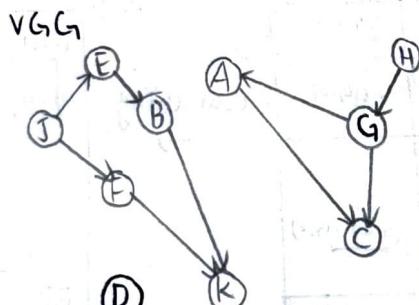
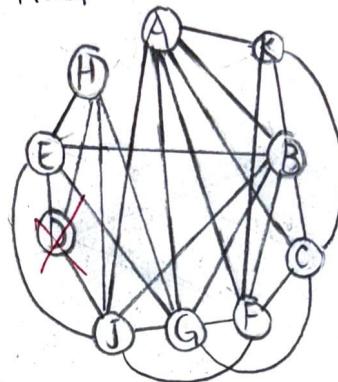
7.

(a)

20

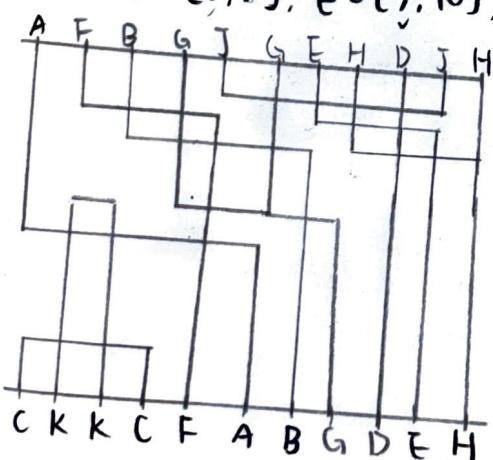


HCG



(b) Yes, 因為 VGG 沒有 cycle
照左 x 由小到大排序

$$A = [1, 6], C = [1, 4], K = [2, 3], F = [2, 5], B = [3, 7], G = [4, 8] \\ J = [5, 10], E = [7, 10], H = [8, 11], D = [9, 9]$$



Track 1: can't route(A, C, K, F, B, G)

Route J

Track 2: can't route(A, C, K), Route F, E

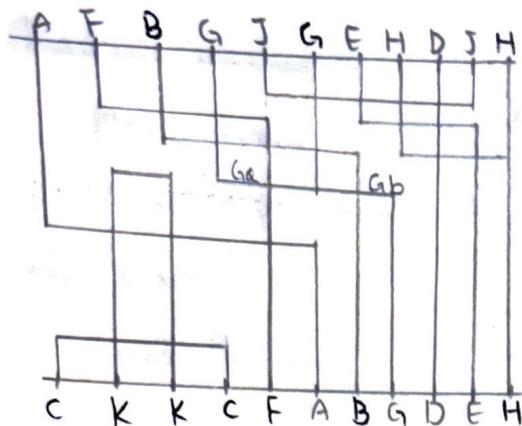
Track 3: can't route(A, C, K), Route B, H

Track 4: can't route(A, C), Route K, G, D

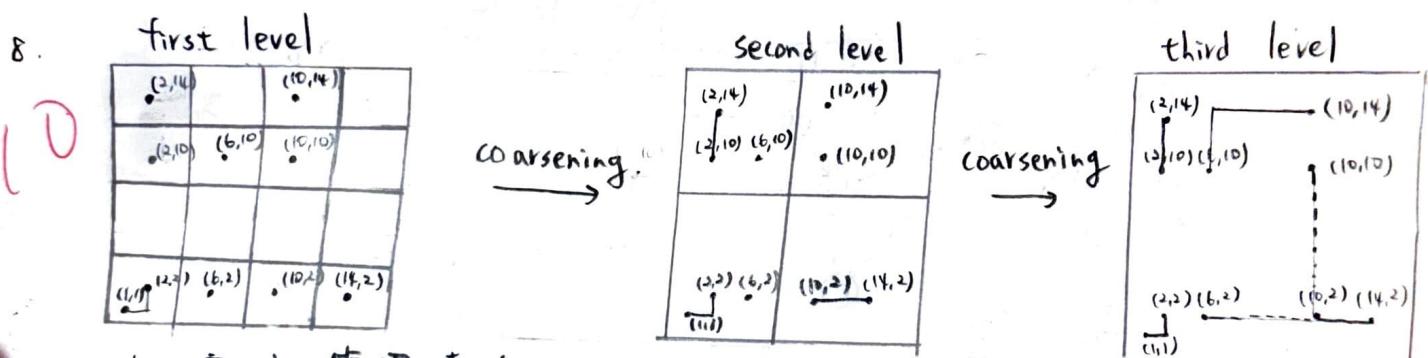
Track 5: Route A.

Track 6: Route C

(C)



G 可以用 dogleg 形成兩段。
但 G-J 间的 channel density 遇到 b
區間
因此 track 最小還是只能到 (b)
的 6 位 track.



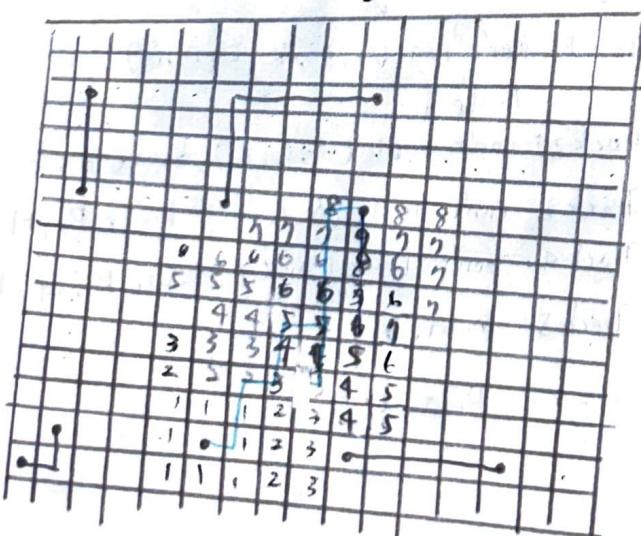
先分割成小範圍繞線，此時只能繞 straight / L shape

後面再合併成一整個考慮繞線。

uncoarsening

發現 (6,2) (10,10) fail 的 connecting

改用 maze routing.



5.

原本的 maze routing 只利用 wave propagation 來尋找

shortest path, 若是要找 shortest path with minimum number of bends 則在 grid 上面加上 bend weight (或是在 cost function 產生 bend penalty).

當路徑要轉彎時，就會使得轉彎的 weight 也沒轉彎的更大，來抑制 bend 的產生。

5. 修改 Lee's 或 Hadlock's 演算法。
① 額外為每個 grid 記錄 (從 source 到該 grid 之最少 bend 及 routing 方向)

完成演算法後即可取得 bend 數與可 back track 的路徑。

∴ 每個 cell 到 source 的 bend 數量最小值可使用 dp 計算

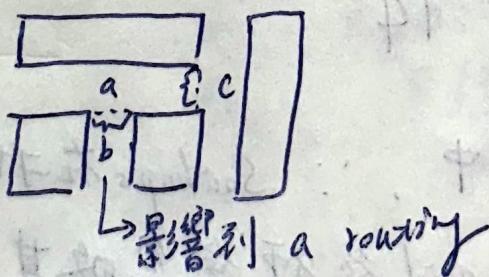
Extend maze routing:
queue $q \leftarrow \text{source}$: \rightarrow 且找到 target
while ($!q.\text{empty}()$) {
 该 cell bend 數 ($dist = n$)
 $\leftarrow \min(\text{dist} = n - 1 \text{ 相鄰 cell}$
 c bend 數 + routing 方向是相
 同 [$(\text{不同} + 1, \text{相同} + 0)$] };

6.

Not correct, the order is $b \rightarrow a \rightarrow c$

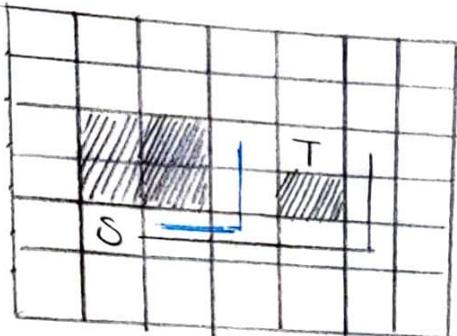
② channel routing 需確保兩側 pin 位置已經 define 好，

若先嘗試 route a , 則缺少 b 處 pin 的位
置資訊。



4.

10



如果用 Soukup algorithm,

路徑會是黑色的 (6)

而用 Lee algorithm 的話，

應該會得到藍色的 (4). $\Rightarrow 6 > 4$.

5. ①

○ 可以藉由選擇離中心點較遠的點作為 start point.

② 使用 double-fan-out.

6.

() 應該是以 slicing 的順序來決定誰先繞 (先切的後繞).

slicing 順序為: $c \rightarrow a \rightarrow b$.

\Rightarrow 繞的順序為 $b \rightarrow a \rightarrow c$.