Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Supplementary materials for

Jie YANG, Kai QIAO, Jian CHEN, Chen CHEN, Lixiang GUO, Bin YAN, 2025. A review of automatic schematic generation techniques and their application to printed circuit boards. *Front Inform Technol Electron Eng*, 26(9):1534-1550.
https://doi.org/10.1631/FITEE.2400612

Algorithm S1 Simulated annealing algorithm based on the logical layout

Input: Initial temperature T_0, termination temperature T_f, temperature decline rate r, number of iterations at each temperature l, initial row sequence state P_0, component set U.

Output: Logical layout result L.

$T = T_0, P = P_0$ ▷ Initialize parameters.
$Count_{NC} = \text{countCrossings}(P_0)$ ▷ Initialize the optimal number of wire crossings.
$P_{\text{best}} = P$ ▷ Initialize the optimal state.
while $T > T_f$ **do**
  for $i = 1$ **to** l **do**
    $P_{\text{new}} = \text{randomSwapRow}(U)$
    $\text{NewCount}_{NC} = \text{countCrossings}(P_{\text{new}})$ ▷ Calculate the number of crossings in new state.
    $\Delta = \text{NewCount}_{NC} - \text{Count}_{NC}$
    if $\Delta < 0$ or $\exp(-\frac{\Delta}{T})$ **then**
      $P = P_{\text{new}}$
      $P_{\text{best}} = P$
      $\text{Count}_{NC} = \text{NewCount}_{NC}$
    end if
  end for
  $T = T \times r$
end while
$L = \text{layout}(P)$ ▷ Form a layout based on State.
return L

Algorithm S2 Layout algorithm based on reinforcement learning

Input: Maximum training episodes Ep, learning rate l_r, netlist H.
Output: Logical layout result L.

Initialize neural network Agent.
Initialize the layout environment Env.
Define the reward function R.
repeat
    Reset(Env) ▷ Reset layout environment.
    S = getInitialState(Env) ▷ Get initial state.
    while not done **do**
        S = embedFeatures(S, Env) ▷ Embedded building block features.
        A = selectAction(Agent)
        done, S, r = envExecute(A) ▷ Execute layout action, update state.
        if *done* **then**
            Update Agent.
        end if
    end while
until $e > Ep$
L = layout(S) ▷ Form a layout based on State.
return L

Algorithm S3 A* algorithm

Input: Route starting point N_0, route ending point N_f.
Output: The routing path p from N_0 to N_f.

Initialize Routing Grid Map G.
Initialize the set of nodes to be traversed O and the set of nodes already traversed C.
aCost[N_0] = 0 ▷ Initialize the actual cost from N_0 to the current point N_c.
eCost[N_0] = getManhattanCost(N_0) × 0.8 ▷ Initialize the estimated cost from N_c to N_f.
Parent[N_0] = null
O.add(N_0, aCost[N_0] + eCost[N_0])
while $O \neq \emptyset$ **do**
    N_c = getMinCostNode(O)
    if $N_c == N_f$ **then**
        return reconstructPath(N_c)
    end if
    for N_i in getNeighbors(N_c, G) **do**
        if $N_i \in C$ **then**
            continue
        end if
        tCost[N_i] = aCost[N_c] + getaCost(N_c, N_i)
        if $N_i \notin O$ or tCost[N_i] < aCost[N_i] **then**
            aCost[N_i] = tCost[N_i]
            eCost[N_i] = getManhattanCost(N_i) × 0.8
            Parent[N_i] = N_c
            O.add(N_i, aCost[N_i] + eCost[N_i])
        end if
    end for
end while
return null ▷ Return null if no path is found at the end of the loop.