Optimal Joining Of Compacted Cells

Andrew Lim

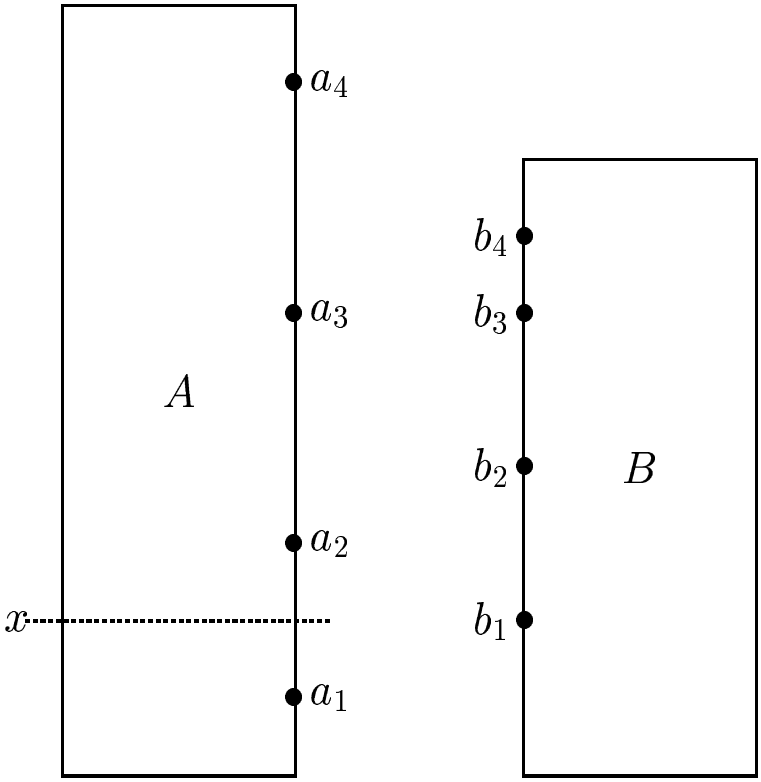
University of Florida

Siu-Wing Cheng

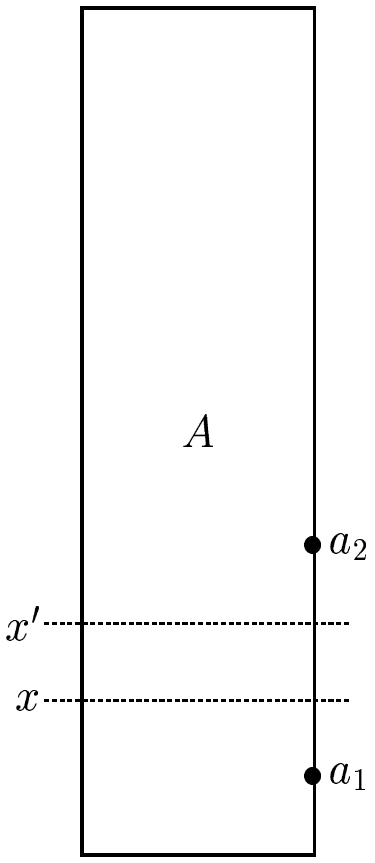
University of Minnesota

Sartaj Sahni

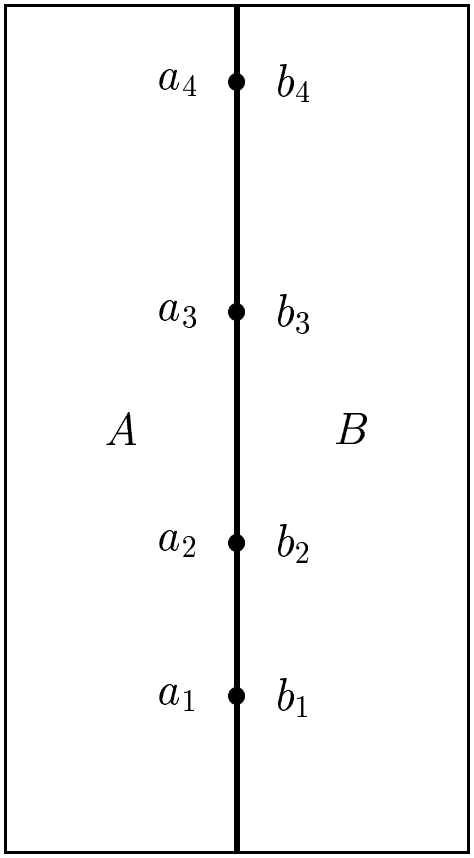
University of Florida



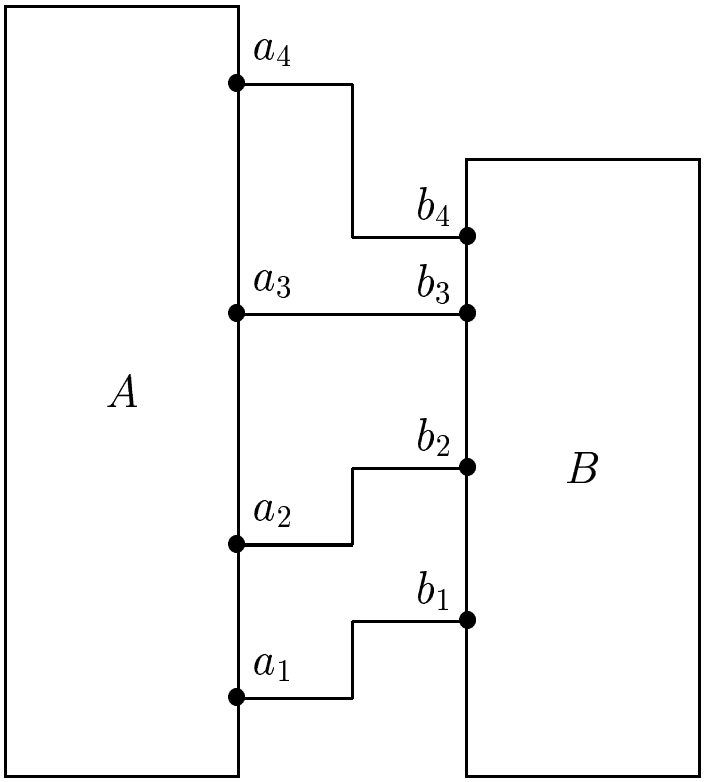
(a) Two adjacent cells



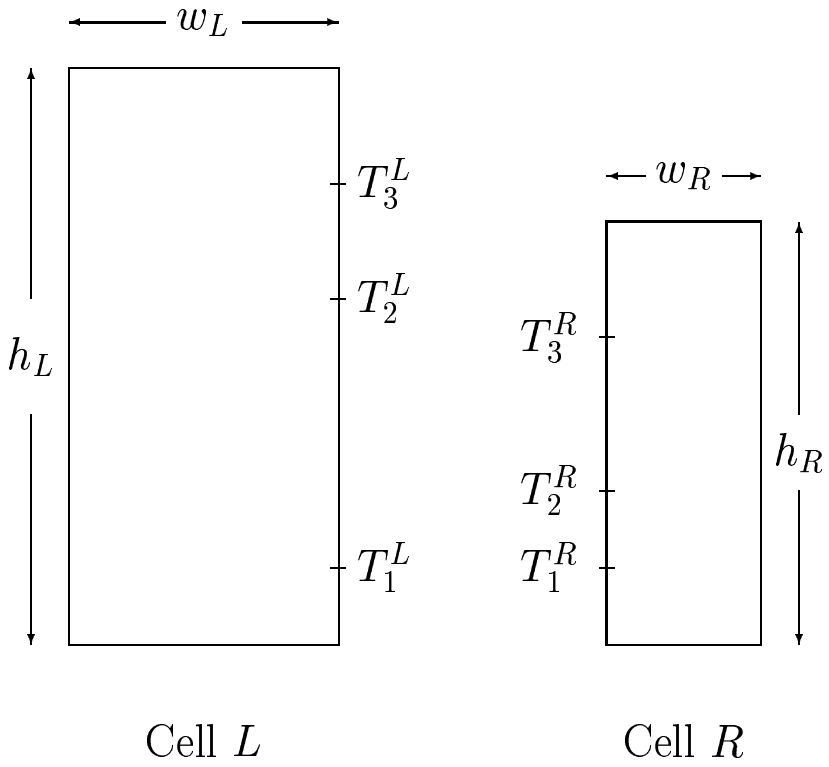
(b) stretching at x



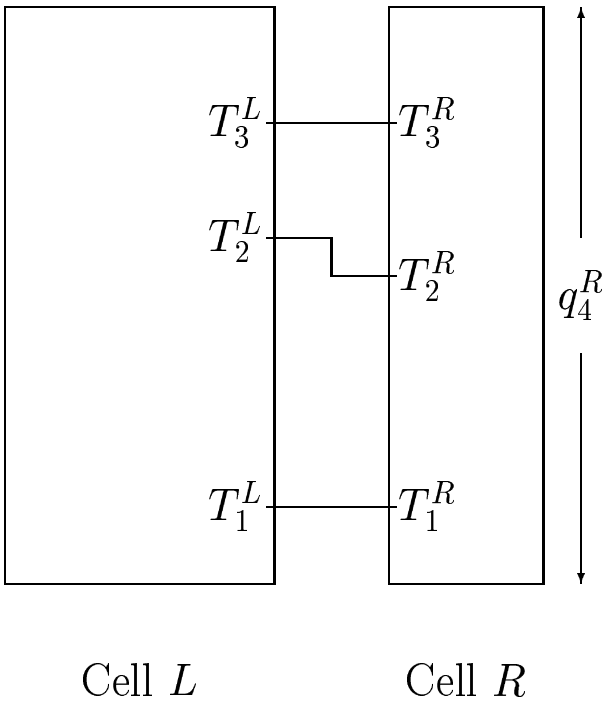
(c) joining by stretching



(d) joining by river routing



(a) Two compacted Cells



(b) After stretching and routing

```
procedure MinHeight(s);
    /* Determine a minimum height legal stretching Q
       that can be routed using s tracks */
begin
    q_0^L := 0; q_0^R := 0;
    for i:= 1 to s do begin
        q_i^L := q_{i-1}^L + Δ_{i-1}^L;
        q_i^R := q_{i-1}^R + Δ_{i-1}^R;
    end;
    for i:= s + 1 to n do begin
        q_i^L := q_{i-1}^L + Δ_{i-1}^L;
        q_i^R := q_{i-1}^R + Δ_{i-1}^R;
        if q_i^R < q_{i-s}^L + s then
            q_i^R := q_{i-s}^L + s
        else if q_i^L < q_{i-s}^R + s then
            q_i^L := q_{i-s}^R + s;
    end;
end;
```

Figure 1: Procedure to find minimum height legal stretch

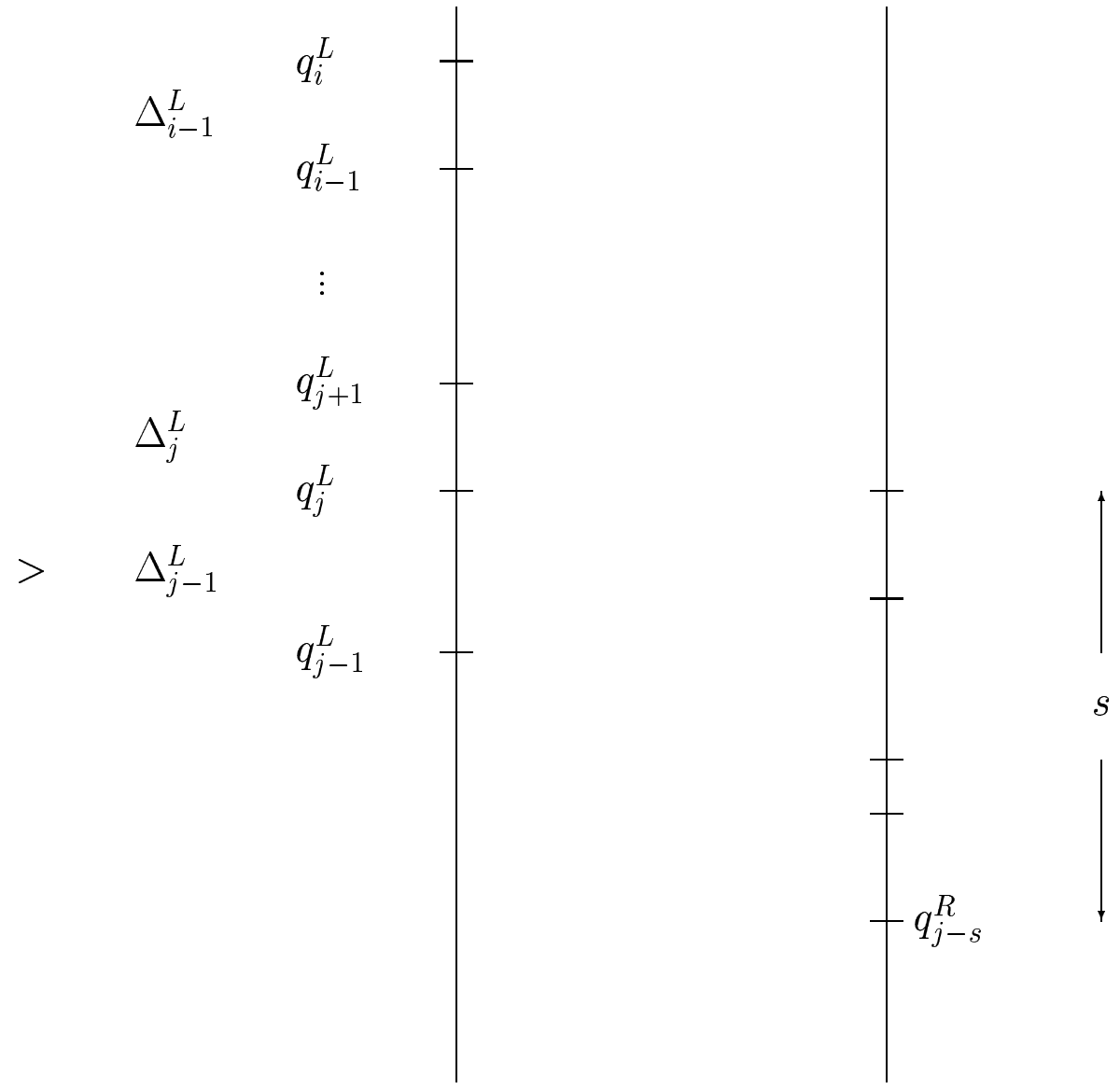


Figure 2: Relationship

procedure *FixedLabeling(Q)*;
/* label some terminals as fixed. The labeled terminals
satisfy conditions $C1 - C4$ */
begin
Initialize a queue to be empty;
if $q_n^L + \Delta_n^L \geq q_n^R + \Delta_n^R$ **then** label T_n^L fixed and add to the queue;
if $q_n^R + \Delta_n^R \geq q_n^L + \Delta_n^L$ **then** label T_n^R fixed and add to the queue;
while queue not empty **do begin**
delete a terminal, say T_i^A, from the queue;
find least j such that $q_{a+1}^A - q_a^A = \Delta_a^A, j \leq a < i$;
label T_a^A, $j \leq a < i$, fixed;
if $j > s$ **then begin**
let $B = \{L, R\} - A$;
add T_{j-s}^B to the queue;
label $T_{j-s}^B, \ldots, T_j^B$ fixed;
end;
end;
end;

Figure 3: Procedure to label fixed terminals

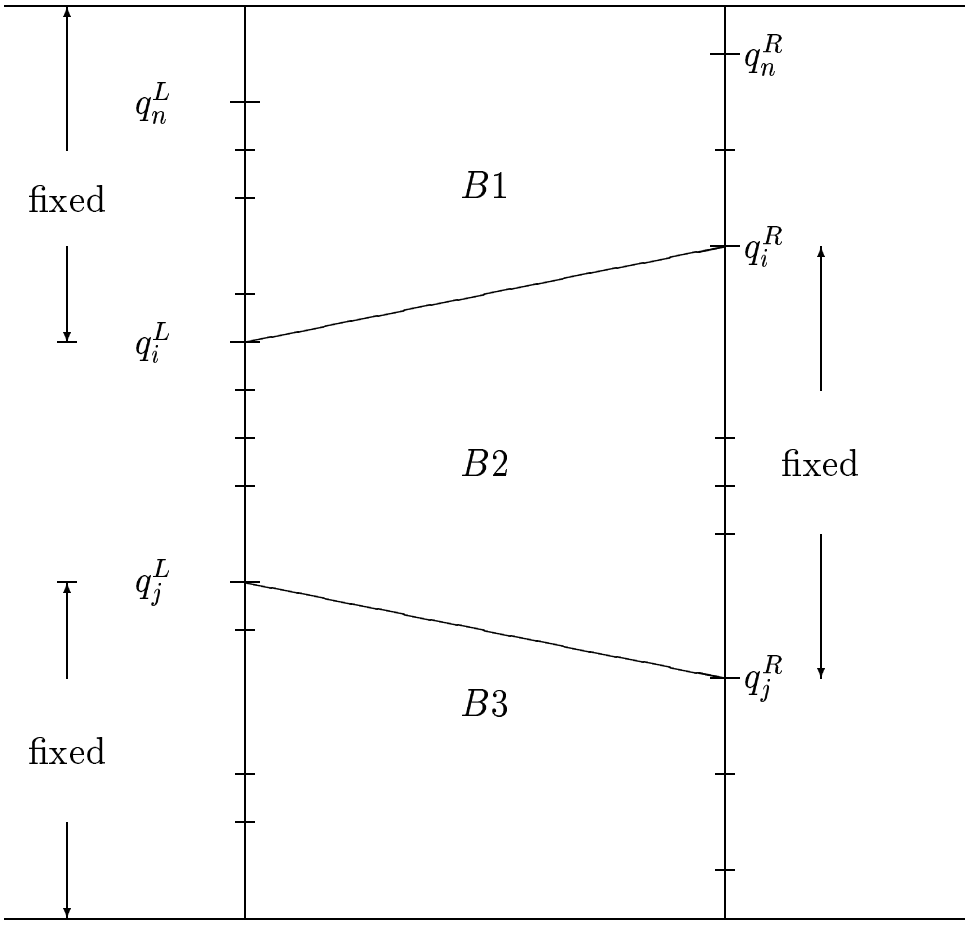


Figure 4: Block Decomposition

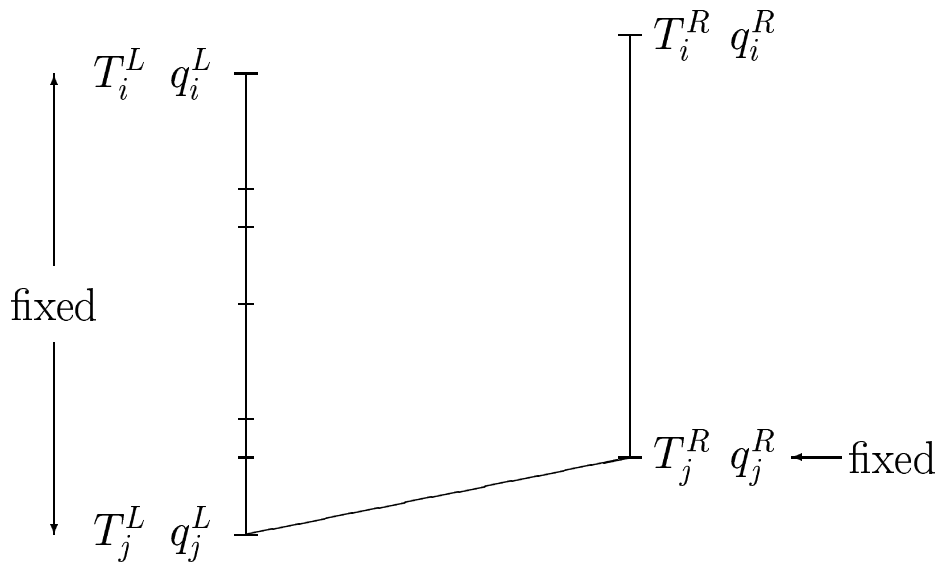


Figure 5: A block with left terminals fixed

procedure *LeftMaxL(i,j)*;
    /* Determine the minimum maximum wire length for
       a block with fixed terminals on the left side. The
       terminals in the block are T_a^A, $A \in \{L, R\}$, $j \leq a \leq i$ */
begin
    if $i = n$ **then** $q_{n+1}^R = MinHeight$;
    $high := q_{i+1}^R$;
    $MaxL := 0$; /* length of longest wire */
    for $a := i$ **downto** $j + 1$ **do**
        if $q_a^L + MaxL < q_a^R$
        then begin /* wire a is longer than $MaxL$ */
            $MaxL := q_a^R - q_a^L$;
            $high := min\{high - \Delta_a^R, q_{a+s}^L - s\}$;
        end
        else begin /* wire a may be longer than $MaxL$ */
            $q_a^R := min\{q_a^L + MaxL, q_{a+1}^R - \Delta_a^R, q_{a+s}^L - s\}$;
            $high := min\{high - \Delta_a^R, q_{a+s}^L - s\}$;
            if $q_a^L - q_a^R > MaxL$
                then $q_a^R := min\{high, q_{a+1}^R - \Delta_a^R, q_a^R + \lceil (q_a^L - q_a^R - MaxL)/2 \rceil\}$;
            $MaxL := max\{MaxL, q_a^L - q_a^R\}$;
        end;
end;

Figure 6: Determine minimum maximum length

procedure *LeftRelocate(i,j)*;

/* Relocate the right terminals of a block whose left terminals are fixed. The maximum wire length is to be $MaxL$ and the terminals are T_a^A, $A \in \{L, R\}$, $j < a \leq i$} */

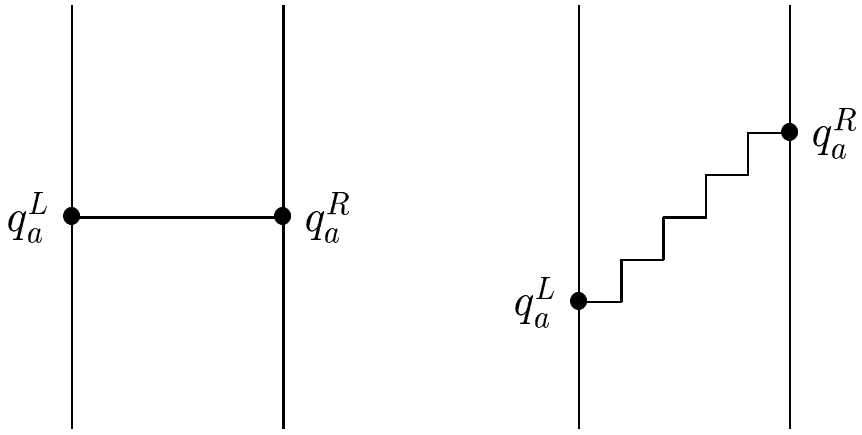
begin

for a:=i **downto** $j+1$ **do**

$q_a^R := min\{q_a^L + MaxL, q_{a+1}^R - \Delta_a^R, q_{a+s}^L - s\}$

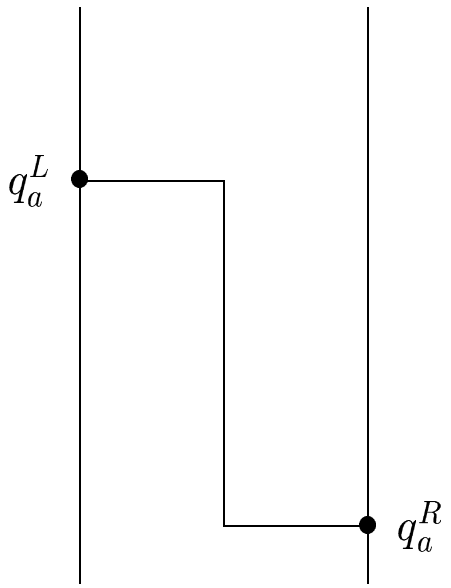
end;

Figure 7: Relocate the right moveable terminals



$q_a^L = q_a^R$ $\qquad$ $q_a^L < q_a^R$

(a) Positive terminals



$$q_a^L > q_a^R$$

(b) Negative terminals

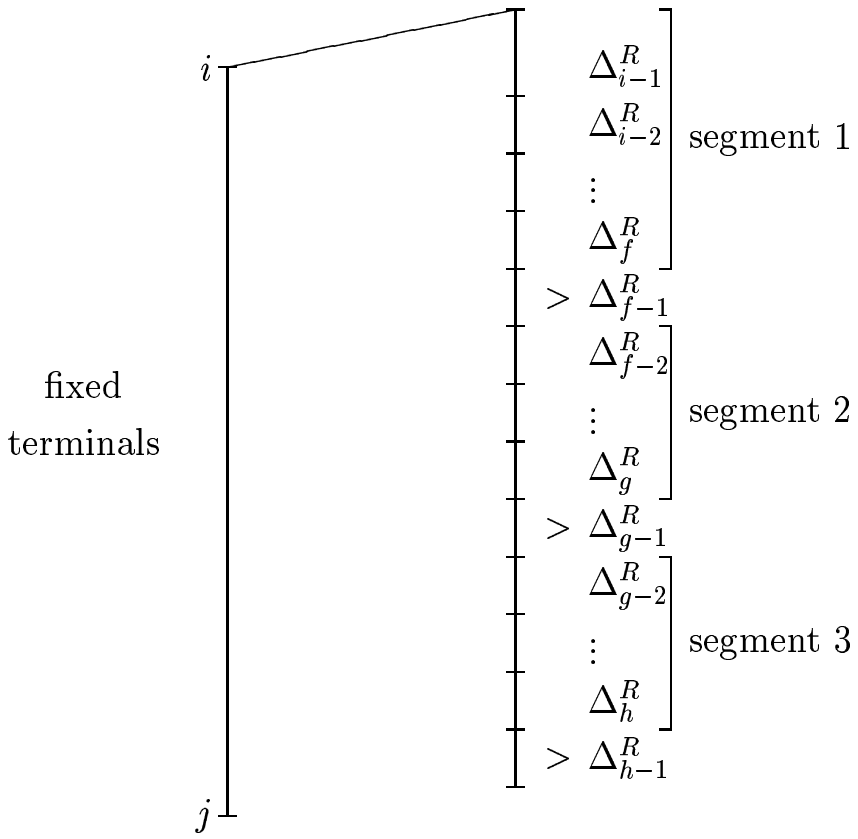


Figure 8: Segments

```
procedure Update(i,j);
    /* Use Offsets to compute q_a^R's */
begin
Initialize an empty stack;
AddValue := 0; EndIndex := i
for a := j to i do
    begin
        p := Offset(a);
        while p <> nil do /* get offsets from the list */
        begin
            get(x, SegBegin) from node p;
            p := next node on the offset list;
            AddToStack((AddValue, EndIndex));
            AddValue := AddValue + x;
            EndIndex := SegBegin;
        end;
        q_a^R := q_a^R + AddValue;
        if (a = EndIndex) and (a < i)
        then DeleteFromStack((AddValue, EndIndex));
    end;
end;
```

Figure 9: Second Pass of total wire length algorithm