Algorithm 1 Heuristic for ETSPTW–MCR

```
Require: Graph $G = (V \cup \{0\}, E)$ with distances $d_{ij}$, time windows $[e_i, \ell_i]$
Require: Charging stations $F \subseteq V \cup \{0\}$ with depot $0 \in F$
Require: Battery capacity $Q$, rate $h$, charging rates $g_i$
Ensure: Feasible route $R$ or infeasible
procedure Heuristic
    $T \leftarrow \mathrm{MST}(G)$ rooted at 0
    $R \leftarrow$ PreorderWalk($T$); append depot 0 to the end
    Phase 1: make route time-window feasible
    for $k \leftarrow 1$ to $|R| - 1$ do
        compute arrival $a_k$ at $R_k$
        if $a_k > \ell_{R_k}$ then                          ▷ late → shift customer forward
            remove $u \leftarrow R_k$
            $j \leftarrow$ FindEarliestPosition($u, k, R$)
            if $j$ = None thenreturn infeasible
            else
                insert $u$ at position $j$
        else if $a_k < e_{R_k}$ then
            wait until $e_{R_k}$
    Phase 2: make route battery-feasible
    $b \leftarrow Q$                                         ▷ residual battery
    for $k \leftarrow 1$ to $|R| - 1$ do
        $e \leftarrow h \cdot d_{R_k, R_{k+1}}$                        ▷ energy to next vertex
        if $b < e$ then                                      ▷ cannot reach $R_{k+1}$
            $s \leftarrow$ FindInsertableStation($k, R, b$)
            if $s$ = None then return infeasible
            else
                insert $s$ at position $k + 1$ in $R$
                $b \leftarrow Q$                             ▷ full recharge
                continue                         ▷ re-evaluate hop after insertion
        $b \leftarrow b - e$                         ▷ consume energy and move on
    return $R$
```

Algorithm 2 FindInsertableStation(k, R, b)

Require: Index k ($1 \le k < |R|$), current route R, residual battery b
Ensure: Station s to insert after R_k or None

```
function FindInsertableStation$(k, R, b)$
    $best \leftarrow$ None,  $bestDetour \leftarrow \infty$
    $i \leftarrow R_k$, $j \leftarrow R_{k+1}$
    $slack \leftarrow tailSlack[k]$                    ▷ cumulative spare time from $k$ onward
    for all $s \in F \setminus R$ do                     ▷ stations not yet on the tour
        $e_1 \leftarrow h \cdot d_{i,s}$                          ▷ energy depot $R_k \rightarrow s$
        if $b < e_1$ then continue
                                                   ▷ cannot even reach $s$
        $extra \leftarrow d_{i,s} + d_{s,j} - d_{i,j}$                 ▷ detour distance
        if $extra > slack$ then continue
        if $extra < bestDetour$ then
            $best \leftarrow s$;   $bestDetour \leftarrow extra$
    return $best$
```

Algorithm 3 FindEarliestPosition(u, k, R)

Require: Customer u was removed from index k of route R
Ensure: Smallest $j > k$ that keeps every arrival $\le \ell$, or None

```
function FindEarliestPosition$(u, k, R)$
    $prev \leftarrow R_k$                                  ▷ vertex before the gap
    $clock \leftarrow arrivalTime(prev)$
    $slack \leftarrow tailSlack[k]$                       ▷ spare time downstream
    for $j \leftarrow k + 1$ to $|R|$ do                    ▷ single pass over suffix
        $next \leftarrow R_j$
        $detour \leftarrow d_{prev,u} + d_{u,next} - d_{prev,next}$
        $t_u \leftarrow clock + d_{prev,u}$                            ▷ arrival at $u$
        if $t_u < e_u$ then
            $wait \leftarrow e_u - t_u$;    $t_u \leftarrow e_u$
            $detour \leftarrow detour + wait$
        else
            $wait \leftarrow 0$
        if $t_u \le \ell_u \wedge detour \le slack$ then
            return $j$                                 ▷ earliest feasible slot
        $clock \leftarrow clock + d_{prev,next}$                ▷ move on without $u$
        if $clock < e_{next}$ then
            $clock \leftarrow e_{next}$
        $slack \leftarrow \min\big(slack,\ \ell_{next} - clock\big)$
        $prev \leftarrow next$
    return None                                   ▷ no legal position found
```