Algorithm 1 Heuristic for ETSPTW–MCR (Improved)

```
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
    Phase 1: create TSP route
    $R \leftarrow$ PREORDERWALK($T$); append depot 0 to the end
    Phase 2: 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 then return *infeasible*
            else
                insert $u$ at position $j$
        else if $a_k < e_{R_k}$ then
            if $R_k$ is a private station then
                $wait \leftarrow e_{R_k} - a_k$
                $x \leftarrow g_{R_k} \cdot wait$
                charge battery by $x$ during idle time
            wait until $e_{R_k}$
    Phase 3: make route battery feasible
    $b \leftarrow Q$                                          ▷ residual battery
    for $k \leftarrow 1$ to $|R| - 1$ do
        $i \leftarrow R_k$;   $j \leftarrow R_{k+1}$
        $e \leftarrow h \cdot d_{i,j}$                               ▷ energy to next node
        if $b < e$ then
            if $i$ is a private charging station then
                $t_i \leftarrow arrivalTime(i)$
                $x \leftarrow e - b$                                  ▷ missing energy
                $\Delta t \leftarrow x / g_i$
                $t_j \leftarrow t_i + \Delta t + d_{i,j}$
                if $t_j \leq \ell_j$ then
                    $b \leftarrow b + x$              ▷ partial charge only what is needed
                    continue
            $s \leftarrow$ FINDINSERTABLESTATION($k, R, b$)
            if $s =$ NONE then
                $(s_1, s_2) \leftarrow$ FINDTWOSTATIONS($k, R, b$)
                if $s_1 =$ NONE $\vee$ $s_2 =$ NONE then return *infeasible*
                else
                    insert $s_1$ at $k + 1$, $s_2$ at $k + 2$ in $R$
                    $b \leftarrow Q$                    ▷ full charge assumed across both
                    continue
            else
                insert $s$ at $k + 1$ in $R$
                $x \leftarrow Q - b$
                $b \leftarrow b + x$
                continue
        $b \leftarrow b - e$
    return $R$
```

Algorithm 2 FindEarliestPosition(u, k, R)

Require: Customer u was removed from index k of route R
Ensure: Smallest $j > k$ that keeps every arrival $\leq \ell$, 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 $t_u \leq \ell_u \wedge detour \leq slack$ 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
```

Algorithm 3 FindInsertableStation(k, R, b)

Require: Index k $(1 \leq k < |R|)$, 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 4 FindTwoStations(k, R, b)

```
Require: Index $k$ ($1 \le k < |R|$), route $R$, residual battery $b$
Ensure: Two stations $(s_1, s_2)$ to insert after $R_k$ or (None, None)
function FindTwoStations$(k, R, b)$
    $bestPair \leftarrow$ (None, None),  $bestDetour \leftarrow \infty$
    $i \leftarrow R_k$;   $j \leftarrow R_{k+1}$
    $slack \leftarrow tailSlack[k]$
    for all $s_1 \in F \setminus R$ do
        $e_1 \leftarrow h \cdot d_{i,s_1}$
        if $b < e_1$ then continue
        $b_1 \leftarrow Q$                                          ▷ assume full charge at $s_1$
        for all $s_2 \in F \setminus R \setminus \{s_1\}$ do
            $e_2 \leftarrow h \cdot d_{s_1,s_2}$
            if $b_1 < e_2$ then continue
            $e_3 \leftarrow h \cdot d_{s_2,j}$
            if $Q < e_3$ then continue
            $extra \leftarrow d_{i,s_1} + d_{s_1,s_2} + d_{s_2,j} - d_{i,j}$
            if $extra > slack$ then continue
            if $extra < bestDetour$ then
                $bestPair \leftarrow (s_1, s_2)$;   $bestDetour \leftarrow extra$
    return $bestPair$
```