## Electronic Companion A – Nomenclature

**A MILP formulation for the eDARP-FC**

**Indices:**

| | |
|---|---|
| $s$ | stations |
| $l$ | laps |
| $v$ | shuttle vehicles |
| $i$ | pick-up and drop-off nodes |

**Parameters:**

| | |
|---|---|
| $S$ | the number of stations ($s = 0$ denotes the depot, and $s = 1, \dots, S-1$ denote passenger stations, and $\mathcal{S} = \{0, .., S-1\}$) |
| $L$ | an upper bound on the number of laps per vehicle ($\mathcal{L} = \{0, \dots, L-1\}$) |
| $V$ | the number of vehicles ($\mathcal{V} = \{0, \dots, V-1\}$) |
| $t_s$ | the traveling time to station $s \in \mathcal{S}$ from the one preceding it on the circuit |
| $\theta$ | the traveling time required to complete a full lap, i.e., $\theta = \sum_{s=0}^{S-1} t_s$ |
| $n$ | the number of passenger requests |
| $\mathcal{P}$ | set of pick-up nodes ($i = 0, \dots, n-1$) |
| $\mathcal{D}$ | set of drop-off nodes ($i = n, \dots, 2n-1$), the drop-off node of pick-up node $i$ is $n+i$ |
| $s(i)$ | the station of pick-up/drop-off node $i$ (we assume that $s(i) \neq s(n+i)$ and $s(i) \neq 0$) |
| $a_i$ | the passenger earliest pick-up time at node $i \in \mathcal{P}$ |
| $d_i$ | the latest time the passenger(s) can be dropped off at drop-off node $i \in \mathcal{D}$ |
| $\delta(i,j)$ | the direct travel time between nodes $i, j \in \mathcal{P} \cup \mathcal{D}$ |
| $\varepsilon_i$ | the earliest arrival time to drop-off node $i \in \mathcal{D}$ |
| $q_i$ | the number of passengers embarking (disembarking) at pick-up (drop-off) node $i \in \mathcal{P}$ ($i \in \mathcal{D}$) |
| $I(i)$ | an indicator that equals 0 if $s(i+n) > s(i)$ for node $i \in \mathcal{P}$ and 1 otherwise |
| $C$ | a shuttle's passenger capacity |
| $\sigma$ | the service duration for each stop at a passenger station (regardless of the number of requests or passengers served) |
| $H$ | the horizon time |
| $\beta^c$ | battery charging rate (unit of charge per time unit) |
| $\beta^d$ | battery discharge per lap |
| $\beta^m$ | the battery's capacity in units of charge |
| $\alpha_1$ | the weight in the objective function of the total number of laps traveled by all vehicles |
| $\alpha_2$ | the weight in the objective function of the passengers' total excess times |

**Decision Variables:**

| | |
|---|---|
| $x_{slv}$ | equals 1 if vehicle $v \in \mathcal{V}$ stops at station $s \in \mathcal{S}$ on lap $l \in \mathcal{L}$, otherwise 0 |
| $u_{ilv}$ | equals 1 if passengers at pick-up node $i \in \mathcal{P}$ board vehicle $v \in \mathcal{V}$ on lap $l \in \mathcal{L}$, otherwise 0 |
| $w_{lv}$ | equals 1 if vehicle $v \in \mathcal{V}$ performs lap $l \in \mathcal{L}$, otherwise 0 |
| $b_{lv}$ | the amount of time that vehicle $v \in \mathcal{V}$ recharges at the beginning of lap $l \in \mathcal{L}$ |
| $B_{lv}$ | the state of charge of the battery of vehicle $v \in \mathcal{V}$ at the beginning of lap $l \in \mathcal{L}$ (before charging) |
| $T_{slv}$ | the time vehicle $v \in \mathcal{V}$ starts service (pick-up or charging) at station $s \in \mathcal{S}$ on lap $l \in \mathcal{L}$ (if no service is performed it represents the arrival time to the station) |
| $Q_{slv}$ | the amount of passengers aboard vehicle $v \in \mathcal{V}$ when it arrives at station $s \in \mathcal{S}$ on lap $l \in \mathcal{L}$ (before any service is performed) |
| $\tau_i$ | the arrival time to drop-off node $i \in \mathcal{D}$ |

**Lap assignment and scheduling for a given node sequence**

**Indices:**

$j$ item in the sequence

**Parameters:**

$n'$ the number of passenger requests in the sequence
$\Psi$ the given sequence of pickup and drop-off nodes
$\mathcal{P}_\Psi$ subset of items in the sequence representing pickup (drop-off) nodes
$\mathcal{D}_\Psi$ subset of items in the sequence representing drop-off nodes
$\hat{s}_j$ the station associated with item $j$
$a_j$ the passenger earliest pick-up time at pick-up node represented by item $j$
$d_j$ the latest time the passenger(s) can be dropped off at the drop-off node represented by item $j$
$\varepsilon_j$ the earliest arrival time to the drop-off node represented by item $j$
$e_j$ a binary parameter that indicates whether item $j$ is served in the same station as item $j-1$
$\lambda_j$ the cumulative number of laps completed since travelling from the depot up to the station associated with item $j$
$t_j$ the traveling time to item $j$ from item $j-1$
$q_j$ the number of passengers embarking/disembarking at item $j$
$\mathcal{R}$ the subset of pickup nodes to which the vehicle arrives with no passengers on board (recharging opportunities)
$\mathcal{R}^0$ the subset of pickup nodes to which the vehicle arrives with no passengers on board (recharging opportunities) and passes anyway by the recharging depot on its way to these nodes
$\mathcal{R}^+$ the subset of pickup nodes to which the vehicle arrives with no passengers on board (recharging opportunities) and may bypasses the pickup node in order to travel specially to the recharging depot

**Decision Variables:**

$y_j$ equals 1 if a type (+) recharging opportunity is taken in item $j = 0, \dots, 2n' - 1$, otherwise 0
$G_j$ the accumulative recharging time up to item $j = 0, \dots, 2n' - 1$ in the sequence ($G_{-1} = 0$)
$T_j$ the time the vehicle arrives to item $j = 0, \dots, 2n' - 1$ in the sequence ($T_{-1} = 0$)

**Dynamic Program**

**State Variables:**

| | |
|---|---|
| $j$ | item in the sequence |
| $T$ | the time at which service begins at that item |
| $B$ | the current state of charge |
| $h$ | equals 1 if recharging opportunity of type (+) was used before item $j$, otherwise 0 |

**Functions:**

| | |
|---|---|
| $\mathbb{Z}(j,B)$ | the range of possible positive integral recharging durations, when the vehicle is in state $(j,T,B,h)$ |
| $z^L(j,B)$ | Lower boundary of $\mathbb{Z}(j,B)$ |
| $z^U(j,B)$ | Upper boundary of $\mathbb{Z}(j,B)$ |
| $next(j)$ | denotes the recharging opportunity that follows recharging opportunity $j$ |
| $F(j,T,B,h)$ | the minimal weighted sum of the total excess times of passengers served in items $j+1,\ldots,2n'$ and the number of laps traveled in order to serve them, given that the vehicle is in state $(j,T,B,h)$ |

## Electronic Companion B – Valid inequalities

Constraints (B.1) to (B.12) below can be applied to tighten the formulation proposed in Section 2.1. Some of these constraints make use of auxiliary binary variables $y_{iv}$ that equal 1 if a request $i \in \mathcal{P}$ is served by vehicle $v$, and 0 otherwise. Furthermore, let $l_i = \left\lceil \frac{d_i}{\left(\theta+\sigma+\frac{\beta^d}{\beta^c}\right)} \right\rceil - 1$ be the latest lap of any vehicle in which a drop-off node $i \in \mathcal{D}$ can be served.

$$\sum_{l=0}^{L-1} u_{ilv} = y_{iv} \qquad \forall i \in \mathcal{P}, \forall v \in \mathcal{V} \qquad \text{(B.1)}$$

$$\sum_{v=0}^{i} y_{iv} = 1 \qquad \forall i \in \mathcal{P}: i \leq V-1 \qquad \text{(B.2)}$$

$$\sum_{l=l_i+1}^{L-1} \sum_{v=0}^{V-1} u_{i-n,l,v} = 0 \qquad \forall i \in \mathcal{D}: s(i-n) < s(i) \qquad \text{(B.3)}$$

$$\sum_{l=l_i}^{L-1} \sum_{v=0}^{V-1} u_{i-n,l,v} = 0 \qquad \forall i \in \mathcal{D}: s(i-n) > s(i) \qquad \text{(B.4)}$$

$$T_{slv} \geq l \cdot \left(\theta + \sigma + \frac{\beta^d}{\beta^c}\right) + \sum_{s'=1}^{s} t_{s'} \qquad \forall s \in \mathcal{S}; \forall l \in \mathcal{L}; \forall v \in \mathcal{V} \qquad \text{(B.5)}$$

$$T_{0lv} \geq l \cdot (\theta + \sigma) + \beta^c \sum_{l'=0}^{l-1} b_{l'v} \qquad \forall l \in \mathcal{L}; \forall v \in \mathcal{V} \qquad \text{(B.6)}$$

$$x_{slv} \leq \sum_{i \in \mathcal{P}: s(i)=s} u_{ilv} + \sum_{\substack{i \in \mathcal{P}: s(i+n)=s \\ \wedge\, s(i)<s(i+n)}} u_{ilv} + \sum_{\substack{i \in \mathcal{P}: s(i+n)=s \\ \wedge\, s(i)>s(i+n)}} u_{i,l-1,v} \qquad \forall s \in \mathcal{S}: s > 0, \forall l \in \mathcal{L}: l > 0, \forall v \in \mathcal{V} \qquad \text{(B.7)}$$

$$x_{s0v} \leq \sum_{i \in \mathcal{P}: s(i)=s} u_{i0v} + \sum_{\substack{i \in \mathcal{P}: s(i+n)=s \\ \wedge\, s(i)<s(i+n)}} u_{i0v} \qquad \forall s \in \mathcal{S}: s > 0, \forall v \in \mathcal{V} \qquad \text{(B.8)}$$

$$\sum_{l=0}^{L-1} x_{slv} \leq \sum_{i \in \mathcal{P}: s(i)=s \,\vee\, s(i+n)=s} y_{iv} \qquad \forall s \in \mathcal{S}: s > 0, \forall v \in \mathcal{V} \qquad \text{(B.9)}$$

$$w_{lv} \leq \sum_{i \in P} u_{ilv} + \sum_{i \in \mathcal{P}: s(i)>s(i+n)} u_{i,l-1,v} \qquad \forall l \in \mathcal{L}: l > 0; \forall v \in \mathcal{V} \qquad \text{(B.10)}$$

$$w_{0v} \leq \sum_{i \in \mathcal{P}} u_{i0v} \qquad \forall v \in \mathcal{V} \qquad \text{(B.11)}$$

$$w_{lv} \leq \sum_{s=0}^{S-1} x_{slv} \qquad \forall l \in \mathcal{L}; \forall v \in \mathcal{V} \qquad \text{(B.12)}$$

$$y_{iv} \in \{0,1\} \qquad \forall i \in \mathcal{P}; \forall v \in \mathcal{V} \qquad \text{(B.13)}$$

Constraints (B.1) link the auxiliary $y$-variables to the $u$-variables. Constraints (B.2) represent symmetry-breaking constraints by forcing the first request to be assigned to the first vehicle, the second request to first or second vehicle, and so on. Constraints (B.3) and (B.4) denote that the pick-up node $i - n \in \mathcal{P}$ corresponding to drop-off node $i \in \mathcal{D}$ cannot be served in a lap later than lap $l_i$ (or even lap $l_i$ or later when $s(i-n) > s(i)$). Constraints (B.5) impose a lower bound on the arrival time at a station in a particular lap, making use of the minimal time consumed for all previous laps and the travel time from the depot to the station. Similarly, constraints (B.6) impose a lower bound on the arrival time at the depot for a particular lap, depending on the number of laps covered and charging times before that lap. Constraints (B.7) to (B.9) prevent stops at stations when no service should be performed. Similarly, constraints (B.10) to (B.12) prevent the activation of laps when no service is required in those laps. Finally, constraints (B.13) define the domain of the $y_{iv}$ variables.

## Electronic Companion C – Complete proof of Claim 4

**Proof 4:** By carefully examining the structure of the scheduling sub-problem formulation (i.e., the MILP (25)-(36) with the $y_j$ variables fixed to an appropriate value), we show that it is characterized by a constraint coefficient matrix which is totally unimodular (TU). The proof of this property relies on the sufficient condition due to Ghouila-Houri (see Padberg, 1976), by which a matrix A is TU if each collection of its columns can be split into two parts so that the sum of the columns in one part minus the sum of the columns in the other part is a vector with entries only 0, +1 or -1.

Suppose then that we are given a subset $C$ of columns of the constraint coefficient matrix, that is, a subset of decision variables. We next explain how to partition them into two disjoint subsets, $C_1$ and $C_2$, in a manner the satisfies the Ghouila-Houri condition. For this simple algorithm, we refer to any pair of variables denoted by the same letter as having the same "type", e.g., $G_j$ and $G_k$, or conversely $T_j$ and $T_k$, for $j, k = 0, \dots, 2n' - 1$. Then, the partitioning procedure we suggest iterates over $j$, and determines in each iteration a subset to which we assign $G_j$ and $T_j$, if they are included in $C$, based on the subset determined in the preceding iteration. In the first iteration, or in any iteration $j$ following an iteration $j-1$ where no variables were added (i.e., $G_{j-1} \notin C$ and $T_{j-1} \notin C$), we assign the variables by default to $C_1$. Next, for each $j = 1, \dots, 2n' - 1$, we assign new variables by default to the same subset as the one determined in the previous iteration. The only exception to this rule is if in any pair of consecutive iterations $j-1$ and $j$, exactly one variable is added in each iteration, and these variables have different types; in this case, the variable in iteration $j$ is assigned to the opposite subset from the one determined in the preceding iteration.

Note that the resulting partition satisfies, by definition of the algorithm steps, the following properties:

(1) If in any iteration $j$, two variables (i.e., both $G_j$ and $T_j$) are included in $C$, then they are both assigned to the same subset.

(2) If in two consecutive iterations, two variables of the same type (i.e., both $T_j$ and $T_{j-1}$, or conversely, $G_j$ and $G_{j-1}$) are included in $C$, then both variables are assigned to the same subset.

(3) If in two consecutive iterations, exactly one variable in each iteration is included in $C$, and these variables have different types (i.e., $G_{j-1}$ and $T_j$, or conversely, $T_{j-1}$ and $G_j$), then these variables are added to different subsets.

We now claim that for any subset of columns of the constraint coefficient matrix $C$, the partition into $C_1, C_2$ as described above, satisfies the Ghouila-Houri condition. We need to demonstrate that for each row (constraint), the sum of the coefficients of variables in $C_1$ minus the sum of the coefficients of variables in $C_2$ is 0, +1 or -1. For Constraints (26), (27), (33) and (34) this holds trivially, as they include only one variable, having a coefficient of either +1 or -1. For Constraints (28), (31) and (32), note that they include only two variables in each row, with coefficients +1 and -1, and by Property 2, they are placed in the same

subset. Thus, the sum of its coefficients is 0, while the other subset is empty, and so the condition holds. For Constraints (29) and (30), note that each row includes two sets of variables of the same type with consecutive indices, and we next analyze their partition according to the number of variables included in $C$. If exactly one out of the four variables is included in $C$, then the condition is trivially satisfied (similar to Constraint (26)). If exactly two variables out of the four are included and they have the same type, then by Property 2 they are assigned to the same subset, and since they have coefficients +1 and -1, the condition holds (similar to Constraints (28), (31) and (32)). Conversely, if exactly two variables are included and they have different types, then there are two cases to consider: either they are from the same iteration, have different coefficients and by Property 1 they are in the same subset; or they are from consecutive iterations, have the same coefficient and they are assigned to different subsets by Property 3. In either case, the difference between the sum of the coefficients in each subset is 0, and so the condition holds. Next, if exactly three variables are included, then they must be in the same subset for Properties 1-2 to be satisfied simultaneously. In this case, the sum of this subset is either +1 or -1 and the other subset is empty, and so the condition holds. Finally, if all four variables are included, then by Properties 1-2, they are all assigned to the same subset, suggesting the sum of the coefficients in this subset is 0 while the other subset is empty, and so the condition holds. To conclude, the condition holds for all constraints. ■

## Electronic Companion D – Pseudocode of the DP algorithm for the integral lap assignment and scheduling sub-problem

**Main**

$F = \{\}$

$Z = \{\}$

$Calc\_F(-1,0,0,0) + \alpha_1 \lambda_{2n'}$

**Calc_F($j, T, B, h$)**

If $(j, T, B, h)$ in $F$: Return $F(j, T, B, h)$

If $j = 2n'$-1 and $B = \beta^d$:

  If $T \leq d_{2n'-1}$ : Return 0

  Else: Return $\infty$

If $j + 1 \in \mathcal{R}$ and $B < \beta^d$: Return $\infty$

If $j \in D_\Psi$ and $T > d_j$: Return $\infty$

$min\_z = \max\left\{1, \frac{1}{\beta^c} \cdot \left(\beta^d \left(\lambda_{next(j+1)} - \lambda_{j+1}\right) - B\right)\right\}$

$max\_z = \min\left\{\max\left\{0, \frac{1}{\beta^c} \cdot \left(\beta^d\left(\lambda'_{2n} - \lambda_{j+1} + 1\right) - B\right)\right\}, \frac{1}{\beta^c} \cdot (\beta^m - B + \beta^d)\right\}$

If $j + 1 \in R^0$:

  best = $\infty$

  best_z=-1

  For z = min_z to max_z:

    val = $Calc\_F\left(j+1, \max\left\{a_{j+1}, T + \sigma\left(1 - e_j\right) + \sigma h e_j + z + t_{j+1}\right\}, B + z \cdot \beta^c - \beta^d\left(\lambda_{j+1} - \lambda_j\right), 0\right)$

    If val < best: best = val, best_z=z

  val = $Calc\_F\left(j+1, \max\left\{a_{j+1}, T + \sigma\left(1 - e_j\right) + \sigma h e_j + t_{j+1}\right\}, B - \beta^d\left(\lambda_{j+1} - \lambda_j\right), 0\right)$

  If best > val: best = val, best_z = 0

  F($j, T, B, h$) = best

  Z($j, T, B, h$) = best_z

If $j + 1 \in \mathcal{R}^+$:

  $min\_z = \max\left\{\frac{\beta^d}{\beta^c}, min\_z\right\}$

  If $j = -1$: $max\_z = \min\left\{\max\left\{0, \frac{1}{\beta^c} \cdot \left(\beta^d\left(\lambda'_{2n} - \lambda_{j+1}\right) - B\right)\right\}, \frac{1}{\beta^c} \cdot (\beta^m - B)\right\}$

  best = $\infty$

  best_z=-1

  For z = min_z to max_z:

    val = $\alpha_1 + Calc\_F\left(j+1, \max\left\{a_{j+1}, T + \sigma\left(1 - e_j\right) + \sigma h e_j + z + t_{j+1} + \theta\right\}, B + z\beta^c - \beta^d\left(\lambda_{j+1} - \lambda_j + 1\right), 1\right)$

    If val < best: best = val, best_z=z

  Val = $Calc\_F\left(j+1, \max\left\{a_{j+1}, T + \sigma\left(1 - e_j\right) + \sigma h e_j + t_{j+1}\right\}, B - \beta^d \cdot \left(\lambda_{j+1} - \lambda_j\right), 0\right)$

  If best > val: best = val, best_z =0

  F($j, T, B, h$) = best

  Z($j, T, B, h$) = best_z

If $j + 1 \in \mathcal{P}_\Psi \backslash \mathcal{R}$:

  $F(j, T, B, h) = Calc\_F\left(j+1, \max\left\{a_j, T + \sigma\left(1 - e_j\right) + \sigma h e_j + t_{j+1}\right\}, B - \beta^d\left(\lambda_{j+1} - \lambda_j\right), 0\right)$

If $j + 1 \in \mathcal{D}_\Psi$:

  $F(j, T, B, h) = \alpha_2 q_{j+1} \cdot \left(T + \sigma\left(1 - e_j\right) + \sigma h e_j + t_{j+1} - \varepsilon_{j+1}\right) + Calc\_F\left(j+1, T + \sigma\left(1 - e_j\right) + \sigma h e_j + t_{j+1}, B - \beta^d\left(\lambda_{j+1} - \lambda_j\right), 0\right)$

Return $F(j, T, B, h)$

## Electronic Companion E – Insertion Screening-Out Procedures

In this Electronic Companion we describe in details the procedures used to screen out infeasible node sequences prior to their complete evaluation via the DP or the two-stage heuristic solution approaches. In particular, we apply three types of considerations to disregard infeasible sequences, namely, simple time-window considerations, vehicle load considerations and the requirement that no passenger completes a full lap on board a vehicle.

**Time windows:**

Let $er_j$, $lt_j$ and $wt_j$ denote the earliest time service can begin at item $j = 0, \dots, 2n' - 1$, the latest time service can begin at item $j$, and the amount of time the vehicle waits to begin service at item $j$, respectively. The earliest and latest times can be computed in linear time using the following recursive functions:

$$wt_j = \begin{cases} \max\left(0, a_j - \left(er_{j-1} + \sigma\left(1 - e_j\right) + t_j\right)\right) & \forall j: \mathcal{P}_\Psi \\ 0 & \forall j: \mathcal{D}_\Psi \end{cases} \tag{D.1}$$

$$er_j = \begin{cases} er_{j-1} + \sigma\left(1 - e_j\right) + t_j & \forall j: \mathcal{D}_\Psi \\ \max\left\{a_j, er_{j-1} + \sigma\left(1 - e_j\right) + t_j\right\} & \forall j: \mathcal{P}_\Psi \backslash \mathcal{R}^0 \\ \max\left\{a_j, er_{j-1} + \sigma\left(1 - e_j\right) + t_j + \max\left(0, (\lambda_j + 1)\frac{\beta^d}{\beta^c} - \sum_{i=0}^{j} wt_i\right)\right\} & \forall j: \mathcal{R}^0 \end{cases} \tag{D.2}$$

$$lt_j = \begin{cases} lt_{j+1} - \sigma\left(1 - e_{j+1}\right) - t_{j+1} & \forall j: \mathcal{P}_\Psi \\ \min\left\{d_j, lt_{j+1} - \sigma\left(1 - e_{j+1}\right) - t_{j+1}\right\} & \forall j: \mathcal{D}_\Psi \backslash \{2n' - 1\} \end{cases} \tag{D.3}$$

where $\lambda_{-1} = 0, \sigma_{-1} = 0, t_0 = 0, er_{-1} = 0, lt_{2n'-1} = d_{2n'-1}$. In Equation (D.1), the required waiting time at each node is calculated. Specifically, observe that the vehicle will never be required to wait at a drop-off node. Equation (D.2) calculates the earliest time service can begin at node $j$. At a drop-off node, the service time at the preceding node and the travel time from the preceding node are added to the earliest service start time at the preceding node. At a pickup node that does not follow a recharging opportunity we take the maximum between the former term and the earliest pickup time. At a pickup node that follows a recharging opportunity of type (0), we also add the minimal extra time that may be required for charging. While the first two are commonly used in DARP studies, the third is specially developed for the properties the eDARP-FC. Equation (D.3) calculates the latest time service can start a node $j$. For drop-off nodes, this is computed as the latest service start time in the following node minus the service time in node $j$ and the travel time to the following node. For pickup nodes, it is the minimum between the above calculation and the latest drop-off time in node $j$. We note that the earliest and latest service start times are calculated with

the assumption that no recharging opportunities of type (+) are used. Thus, the resulting bounds are valid for any lap assignment.

To conclude, a sequence can be screened out if for any node included in it, the earliest service start time is later than the latest service start time, i.e., if there exists a node $j \in \mathcal{P}_\Psi \cup \mathcal{D}_\Psi$ that satisfies $er_j > lt_j$.

**Vehicle occupancy:**

We denote by $Q_j$ the occupancy of the vehicle, that is, the number of passengers on board, after service at node $j$. This can be computed as follows:

$$Q_j = \begin{cases} Q_{j-1} + q_j & j \in \mathcal{P}_\Psi \\ Q_{j-1} - q_j & j \in \mathcal{D}_\Psi \end{cases}$$

A sequence can be screened out if for any node included in it, the occupancy exceeds the passenger capacity, i.e., if there exists an item $j \in \mathcal{P}_\Psi$ that satisfies $Q_j > C$.

**Passengers do not complete full laps:**

Let $j' \in \mathcal{D}_\Psi$ be the item in the sequence representing the drop-off node of the pick-up node in item $j \in \mathcal{P}_\Psi$. For any item $j \in \mathcal{P}_\Psi$ such that $s_j > s_{j'}$ a feasible solution must satisfy $\lambda_{j'} = \lambda_j + 1$. Similarly, for any item $j \in \mathcal{P}_\Psi$ such that $s_j < s_{j'}$ a feasible solution must satisfy $\lambda_{j'} = \lambda_j$. A sequence can be screened out if it does not satisfy the above conditions for any pick-up items.

## Electronic Companion F – Scheduling Heuristics

As described in Section 3, the complexity of the proposed Dynamic Programming algorithm is $O\left(\left(\frac{\beta^m}{\beta^c}\right)^3 \cdot n'^2\right)$. Namely, as the ratio $\frac{\beta^m}{\beta^c}$ increases or the number of requests in the sequence $n'$ increases, the worst-case runtime may become too long for use in the LNS framework, i.e., permitting only a small number of solution evaluations during a limited running time. For this purpose, we propose a two-stage heuristic that first determines several possible lap assignments, and then determines the schedule for each assignment via an LP (Section 2) or a scheduling heuristic. Following Observation 1, we focus on the recharging decisions, with the understanding that once these decisions are taken, a schedule is fully determined. Below, we present sketches of two scheduling heuristics that are based on the concepts of recharging as early and as late as possible, respectively.

**Recharge as early as possible heuristic**

**Input:** a sequence of pick-up nodes, drop-off nodes and visits to the recharging depot
**Iterate forward** over the visits to the recharging depot
    At each visit, recharge as much as possible without:
        - Violating the latest drop-off time in any of the following drop-off nodes in the sequence
        - Exceeding the maximum battery capacity
        - Exceeding the battery charge that is required to reach the end of the sequence
    If the total amount charged is insufficient to reach the following visit to the recharging depot:
        Break, no feasible recharging schedule exists
**Return** the recharging schedule

**Recharge as late as possible heuristic**

**Input:** a sequence of pick-up nodes, drop-off nodes and visits to the recharging depot
Assign the recharging required to traverse all of the sequence to the last visit to the recharging depot
**Iterate backwards** over the visits to the recharging depot
    At the current visit, check whether there is a time-window violation in any of the following drop-off nodes in the sequence
    If such a time window violation exists:
        Set temp_visit to the current visit
        **While** temp_visit is not the first visit to the recharging depot
            Set temp_visit to the visit that precedes is
            Shift as much recharging time as possible from current visit to temp_visit until either:
                - No time window violation remains
                - There is no waiting in the pick-up nodes between temp_visit and the visit that follows it
            If the time windows violation is resolved: Break
        If the time window violation remains
            **Return** no feasible recharging schedule exists
        Shift the minimal recharging time from current visit to the one that precedes it such that:
            - The battery capacity at current visit is not exceeded (after charging)
            - The battery charge does not exceed the amount needed to reach the end of the sequence
If the battery capacity is exceeded in the first visit
    **Return** no feasible recharging schedule exists
**Return** the recharging schedule