Appendix. (Online)

A. Proofs of Lemmas and Theorems

Proof of Proposition 1 Let c denote the cost to serve both customers based on $\mathbf{v}$. Then,

$$c = d_i f(v_i) + d_j f(v_j) = (\bar{d}_i + \tilde{d}_i) f(v_i) + (\bar{d}_j + \tilde{d}_j) f(v_j).$$

The resulting time to visit both customers is

$$\tilde{T} = \frac{\bar{d}_i + \tilde{d}_i}{v_i} + \frac{\bar{d}_j + \tilde{d}_j}{v_j} = \frac{\bar{d}_i}{v_i} + \frac{\bar{d}_j}{v_j} + \frac{\tilde{d}(v_i + v_j)}{v_i v_j}.$$

If the speed towards customer u_i is decreased on $\tilde{d}_i$ by δ units and increased towards u_j by δ units on $\tilde{d}_j$, the resulting time to serve both customers is

$$\tilde{T}_\delta = \frac{\bar{d}_i}{v_i} + \frac{\tilde{d}_i}{v_i - \delta} + \frac{\tilde{d}_j}{v_j + \delta} + \frac{\bar{d}_j}{v_j} = \frac{\bar{d}_i}{v_i} + \frac{\bar{d}_j}{v_j} + \frac{\tilde{d}(v_i + v_j)}{v_i v_j + (v_i - v_j)\delta - \delta^2}.$$

It follows that $\tilde{T}_\delta < \tilde{T}$, since $0 < \delta < v_i - v_j$.

The cost after changing speeds is

$$c_\delta = \bar{d}_i f(v_i) + \tilde{d}_i f(v_i - \delta) + \bar{d}_j f(v_j) + \tilde{d}_j f(v_j + \delta).$$

It follows that $c_\delta \overset{\text{Obs.1}}{<} c$. □

Proof of Proposition 2 We prove this proposition by contradiction and assume that $t_i < b_i$ holds. Choose $\tilde{d}_i, \bar{d}_i, \tilde{d}_{i+1}, \bar{d}_{i+1}$ as described and choose a δ such that $0 < \delta < v_i - v_{i+1}$ and $t_{i-1} + \frac{\bar{d}_i}{v_i} + \frac{\tilde{d}_i}{v_i - \delta} \leq b_i$. Based on Proposition 1, the total cost of sequence S can be reduced by decreasing the speed on $\tilde{d}_i$ by δ and increasing the speed on $\tilde{d}_{i+1}$ by δ. Note that the total time to traverse these distances also decreases, as shown in Proposition 1. Thus, this transformation does not affect the remaining sequence. However, the cost reduction is a contradiction the assumption that $\mathbf{v}$ has minimal cost. □

Proof of Proposition 3 We prove this proposition by contradiction and assume that $t_i > a_i$ holds. Choose $\tilde{d}_i, \bar{d}_i, \tilde{d}_{i+1}, \bar{d}_{i+1}$ as described and choose a δ such that $0 < \delta < v_{i+1} - v_i$ and $t_{i-1} + \frac{\bar{d}_i}{v_i} + \frac{\tilde{d}_i}{v_i + \delta} \geq a_i$. Based on Proposition 1, we can reduce the total cost of S by increasing the speed on $\tilde{d}_i$ by δ and decreasing the speed on $\tilde{d}_{i+1}$ by δ. Note that the total time to traverse these distances also decreases, as shown in Proposition 1. Thus, this transformation does not affect the remaining sequence. However, the cost reduction is a contradiction the assumption that $\mathbf{v}$ has minimal cost. □

Let $t_i, t_i^*, \tilde{t}_i$ denote service start times at customer u_i based on speed vectors $\mathbf{v}, \mathbf{v}^*$, and $\tilde{\mathbf{v}}$, respectively, where $\mathbf{v} = (v_1, v_2, \ldots, v_k)$ is some optimal speed vector within the bounds $[v_{min}, v_{max}]$, $\mathbf{v}^* = (v_1^*, v_2^*, \ldots, v_k^*)$ denotes the resulting vector output by Algorithm 2 and $\tilde{\mathbf{v}} = (\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_k)$ is the speed vector that is returned by Algorithm 1. For this proof, we assume, without loss of generality, that the service start times $t_i, t_i^*, \tilde{t}_i$ for $u_i \in S$ are set to be as early as possible.

Proof of Proposition 4 We first show the feasibility of $\mathbf{v}^*$ by observing that $v_{min} \leq \max(v_{opt}, \tilde{v}_i) \leq v_{max}$, see the definition of v_{opt}, and vector $\tilde{\mathbf{v}}$ is feasible, see Algorithm 1. Hence, all entries are within the speed bounds. The feasibility of sequence S using vector $\mathbf{v}^*$ is ensured if all customers are served within their time windows, which is achieved by setting the departure time from node u_{i-1} to node u_i as $\tilde{t}_i - d_i/v_i^*$ since $\tilde{\mathbf{v}}$ is feasible and $\tilde{v}_i \leq v_i^*$. □

Proof of Proposition 5 Algorithm 2 returns *infeasible* if Algorithm 1 returns *infeasible*. Hvattum et al. (2013) have shown that Algorithm 1 is correct. It holds that if Algorithm 1 returns *infeasible*, no feasible solution exists to S with speeds within $[0, v_{max}]$. Thus, no feasible solution to S with speeds in the interval $[v_{min}, v_{max}]$ can exist.

Proof of Lemma 1 We prove this lemma by contradiction. Suppose the cost of $\mathbf{v}^*$ are greater than the cost of an optimal speed vector $\mathbf{v}$ for function f. Then, it holds that the two speed vectors differ in at least one entry. From Proposition 4 it follows $t_j^* \leq \tilde{t}_j$ for $j = 1, \ldots, k$ since the service start times are assumed to be minimal and $\tilde{v}_j \leq v_j^*$ for $j = 1, \ldots, k$. It has to hold that $v_{max} - v_{min} > 0$, otherwise $\mathbf{v}^*$ and $\mathbf{v}$ are equal. Consider the first index i for which $v_i^* \neq v_i$.

(1) $v_i^* = v_{opt}$. We distinguish between two cases:

(1.1) $v_i < v_{opt}$. This case leads to a contradiction to the optimality of $\mathbf{v}$ since

* $f(v_i) > f(v_i^*) = f(v_{opt})$ since v_{opt} is unique and
* $t_{i-1}^* = t_{i-1}$ and thus, $t_i^* \leq t_i$.

(1.2) $v_i > v_{opt} = v_i^*$. From the definition of $\mathbf{v}^*$, it follows $v_{opt} \geq \tilde{v}_i$. We provide the proof for $v_{opt} > \tilde{v}_i$, the case $v_{opt} = \tilde{v}_i$ is given in (2). It holds that $t_i < t_i^* \leq \tilde{t}_i$, otherwise v_i can be decreased, which leads to a contradiction to the optimality of $\mathbf{v}$. We distinguish between two cases, namely $i = k$ or $i < k$: In the first case, a contradiction follows since $f(v_k) > f(v_k^*)$ and $t_k < t_k^* \leq \tilde{t}_k$ holds. In the second case, we assume that a smallest index j exists such that $\tilde{v}_j < \tilde{v}_{j+1}$ and $i \leq j$. Based on Proposition 3, it follows that $\tilde{t}_j = a_j$. If $j = i$, then this contradicts the feasibility of $\mathbf{v}$ since it holds $t_i < t_i^* \leq \tilde{t}_i$ holds. Thus, we assume that $i < j$ holds. It follows that $\tilde{v}_i \geq \tilde{v}_{i+1} \geq \cdots \geq \tilde{v}_j < \tilde{v}_{j+1}$. Let l be the smallest index with $i \leq l \leq j$ and $v_l > v_{l+1}$. Suppose there is no such index l. It would follow that $\tilde{\mathbf{v}}_{\tilde{l}} < \mathbf{v}_{\tilde{l}}$ for $\tilde{l} = i, \ldots, j$. However, since $\tilde{t}_j = a_j$ holds, it follows that $t_j \geq \tilde{t}_j = a_j$. Together with the fact that $t_i < t_i^* \leq \tilde{t}_i$ holds, it follows that $\mathbf{v}$ applied to the sequence would result in idle times. This implies that v_i could be decreased, which is a contradiction to the optimality of $\mathbf{v}$. Thus, an index l needs to exist with $i \leq l \leq j$ and $v_l > v_{l+1}$. Based on Proposition 2, it follows that $t_l = b_l$. However, this is a contradiction to the feasibility of $\tilde{\mathbf{v}}$. Thus, no index j exists and $\tilde{v}_i \geq \tilde{v}_{i+1} \geq \cdots \geq \tilde{v}_k$ holds. Based on the definition of $\mathbf{v}^*$, it follows $v_{\tilde{l}}^* = v_{opt}$ for $\tilde{l} = i, \ldots, k$. However, this is a contradiction to the optimality of $\mathbf{v}$ since v_i could be decreased. This concludes case (1).

(2) $v_i^* = \tilde{v}_i$. From the definition of $\mathbf{v}^*$, it follows that $\tilde{v}_i \geq v_{opt}$. We distinguish between two cases:

(2.1) $t_i \geq \tilde{t}_i$. It holds that $t_{i-1} = t_{i-1}^* \leq \tilde{t}_{i-1}$. Suppose $v_i > v_i^* = \tilde{v}_i$ holds. However, this is a contradiction to the optimality of $\mathbf{v}$ since $f(v_i) > f(v_i^*) \geq f(v_{opt})$ holds. It follows that $v_i < v_i^* = \tilde{v}_i$ holds. From here, we distinguish between two cases:

(2.1.1) $t_i = \tilde{t}_i$. It follows that $t_{i-1} < \tilde{t}_{i-1}$. Note that index i must be greater than 1; otherwise, $\mathbf{v}$ is infeasible since $\tilde{\mathbf{v}}$ departs at time 0 from the depot. Suppose $\tilde{v}_{i-1} < \tilde{v}_i$ holds. Based on Proposition 3, it follows $\tilde{t}_{i-1} = a_{i-1}$, which is a contradiction to the feasibility of $\mathbf{v}$ since $t_{i-1} < \tilde{t}_{i-1}$. Therefore, it holds $\tilde{v}_{i-1} \geq \tilde{v}_i$. Based on the definition of $\mathbf{v}^*$, it holds $v_{i-1}^* = \tilde{v}_{i-1}$. Moreover, we assumed that $v_l = v_l^*$ for $l < i$. Thus, $v_{i-1} = v_{i-1}^* = \tilde{v}_{i-1}$. However, this implies $v_{i-1} > v_i$ and based on Proposition 2, it follows $t_{i-1} = b_{i-1}$, which contradicts the feasibility of $\tilde{\mathbf{v}}$.

(2.1.2) $t_i > \tilde{t}_i$. In this case, it holds $t_{i-1} \leq \tilde{t}_{i-1}$, $v_i < \tilde{v}_i$, and $t_i > \tilde{t}_i$. We assume that a smallest index j exists with $\tilde{v}_j > \tilde{v}_{j+1}$ and $i < j$. Based on Proposition 2, it follows $\tilde{t}_j = b_j$. Note that index i and j cannot be equal since $t_i > \tilde{t}_i$ holds. It follows $\tilde{v}_i \leq \tilde{v}_{i+1} \leq \cdots \leq \tilde{v}_j > \tilde{v}_{j+1}$. Let l denote the smallest index with $i \leq l \leq j$ and $v_l < v_{l+1}$. Such an index needs to exist; otherwise $\mathbf{v}$ is infeasible. Based on Proposition 3, it follows $t_l = a_l$. However, this is a

contradiction to the feasibility of $\tilde{\mathbf{v}}$ since based on Algorithm 1 speed vector $\tilde{\mathbf{v}}$ does not contain idle times and $t_i > \tilde{t}_i$ holds. Thus, no index j exists and $\tilde{v}_i \leq \tilde{v}_{i+1} \leq \cdots \leq \tilde{v}_k$ holds. Note that $\tilde{t}_k = b_k$ holds. It follows that an index $\bar{j}$ with $i \leq \bar{j} \leq k$ and $v_{\bar{j}} > v_{\bar{j}+1}$, exists, which leads to the same contradiction as shown.

(2.2) $t_i < \tilde{t}_i$. We distinguish between two cases:

(2.2.1) $v_i < \tilde{v}_i$. It follows that $t_{i-1} < \tilde{t}_{i-1}$. Recall that as in case (2.1.1) index i has to be grater than 1. Assume that there exists a greatest index j with $\tilde{v}_j < \tilde{v}_{j+1}$ and $j \leq i$. Based on Proposition 3, it follows that $\tilde{t}_j = a_j$. Note that index i and j cannot be equal since $t_{i-1} < \tilde{t}_{i-1}$ holds. It follows $\tilde{v}_j < \tilde{v}_{j+1} \geq \cdots \geq \tilde{v}_{i-1} \geq \tilde{v}_i$. Let l be the greatest index with $j \leq l \leq i$ and $v_l > v_{l+1}$. Such an index needs to exist; otherwise $\mathbf{v}$ would be infeasible since $\tilde{\mathbf{v}}$ does not contain idle times, $t_i < \tilde{t}_i$, $\tilde{t}_j = a_j$, and $v_{\bar{j}} < \tilde{v}_{\bar{j}}$ for $\bar{j} = j+1, \ldots, i$. Based on Proposition 2, it follows that $t_l = b_l$. However, this is a contradiction to the feasibility of $\tilde{\mathbf{v}}$. Thus, no index j exists and it follows $\tilde{v}_1 \geq \tilde{v}_2 \geq \cdots \geq \tilde{v}_i$. In order to ensure that $t_{i-1} < \tilde{t}_{i-1}$ holds, it has to exist a greatest index l with $l \leq i$ and $v_l > v_{l+1}$. However, as shown in this case, this leads to a contradiction of $\tilde{\mathbf{v}}$.

(2.2.2) $v_i > \tilde{v}_i$. It follows that $f(v_i) > f(\tilde{v}_i) \geq f(v_{opt})$. This case is the same as (1.2) with the difference that $v_i^* = \tilde{v}_i$ and $f(v_i^*) > f(\tilde{v}_i) \geq f(v_{opt})$. The conclusion is that $\tilde{v}_i \geq \tilde{v}_{i+1} \geq \cdots \geq \tilde{v}_k$ holds. It follows that either $t_k < \tilde{t}_k$ or $v_{\bar{j}} > v_{\bar{j}+1}$ for some index $\bar{j}$ with $i \leq \bar{j} \leq k$. In the first case, the speed of v_i can be decreased since $\tilde{\mathbf{v}}$ is feasible, which is a contradiction to the optimality of $\mathbf{v}$. In the second case, choose $\bar{j}$ minimal. It follows $v_i \leq \cdots \leq v_{\bar{j}} > v_{\bar{j}+1}$. In this case, Proposition 2 requires that $t_{\bar{j}} = b_{\bar{j}}$. However, this necessitates that $\mathbf{v}$ results in idle times, otherwise $\tilde{\mathbf{v}}$ would be infeasible.

This concludes case (2.2) and the proof of this lemma. □

Proof of Claim 1 We prove this claim by contradiction. Suppose $v_i < v_{i+1}$ holds. Choose a distance $\tilde{d}$ with $\tilde{d} = \min\{d_i, d_{i+1}\}$. Based on Proposition 1, it holds the speed towards u_i can be increased by δ on $\tilde{d}$ and the speed towards u_{i+1} can be decreased by δ on $\tilde{d}$. For some δ such that the start of the service at u_i remains feasible and $v_i + \delta \leq v_{i+1} - \delta$. This change of speed is possible since $a_i \neq b_i$ holds. The service start time at u_{i+1} remains feasible by using some idle time. Thus, the adjusted speed vector is feasible and leads to smaller cost, which is a contradiction. □

Proof of Claim 2 We prove this claim by contradiction. Suppose $v_i > v_{i+1}$ holds. Choose a distance $\tilde{d}$ with $\tilde{d} = \min\{d_i, d_{i+1}\}$. Based on Proposition 1, it holds the speed towards u_i can be decreased by δ on $\tilde{d}$ and the speed towards u_{i+1} can be increase by δ on $\tilde{d}$. For some δ such that the start of the service at u_i remains feasible and $v_i - \delta \geq v_{i+1} + \delta$ holds. This change of speed is possible since $a_i \neq b_i$ holds. The service start time at u_{i+1} remains feasible by using some idle time. Thus, the adjusted speed vector is feasible and leads to smaller cost, which is a contradiction. □

Proof of Proposition 6 Let $\mathbf{v} = (v_1, \ldots, v_k)$ denote the cost-minimal speed vector. We distinguish between two cases, either S with $\mathbf{v}$ contains a customer with $t_i = a_i$ or not.

(1) Let u_l denote the last customer in S with $t_l = a_l$. If $a_l \neq b_l$, then it follows $v_l \leq v_{l+1}$ since $\mathbf{v}$ is cost-minimal, see Claim 2. Suppose $l = k$, then $\tilde{S}(\mathbf{v}) = (u_1, \ldots, u_k)$ follows. Otherwise, select $\tilde{S}(\mathbf{v}) = (u_1, \ldots, u_l)$ and $\bar{S}(\mathbf{v}) = (u_{l+1}, \ldots, u_k)$. Suppose $v_i < v_{i+1}$, with $l < i < k$. Based on Proposition 3, $t_i = a_i$ follows, which is a contradiction to the assumption that u_l is the last customer with $t_l = a_l$. Thus, it follows that $v_i \geq v_{i+1}$ for $l < i < k$. Suppose $t_i = a_i$ for some $i > l$. This is a contradiction to the assumption that u_l is the last customer with $t_l = a_l$.

(2) Select $\bar{S}(\mathbf{v}) = (u_1, \ldots, u_k)$. Equally to case (1), sequence $\bar{S}(\mathbf{v})$ does not contain two consecutive customers with $v_i < v_{i+1}$ and thus, $v_i \geq v_{i+1}$ for $0 < i < k$ holds.

The number of sub-sequences $\bar{S}_i(\mathbf{v}), i = 1, \ldots, p$, should be minimal with the restriction that all customers within such a sub-sequence have equal speeds.

□

Proof of Claim 3 Let $\mathbf{v}^* = (v_1^*, v_2^*, \ldots, v_k^*)$ denote the optimal speed vector of the input. Based Proposition 6, there is a matching partitioning for $\mathbf{v}^*$.

In the following, we consider an iteration of Algorithm 3 and show that all vectors added to $\mathcal{V}$ have a matching partitioning. Let $\mathbf{v} = (v_1, v_2, \ldots, v_k)$ denote the speed vector of the beginning of this iteration. The partitioning of $\mathbf{v}$ is given as $\tilde{S}(\mathbf{v})$ and $\bar{S}_i(\mathbf{v}), i = 1, \ldots, p$. Let $\mathbf{v}' = (v_1', v_2', \ldots, v_k')$ denote the speed vector of the end of this iteration. Let $\hat{\mathbf{v}} = (\hat{v}_1, \ldots, \hat{v}_j, \hat{v}_{j+1}, \ldots, \hat{v}_k)$ denote some speed vector that is added to $\mathcal{V}$ in this iteration. It should hold that $\hat{\mathbf{v}} \neq \mathbf{v}$ and $\hat{\mathbf{v}} \neq \mathbf{v}'$.

At the beginning of an iteration, a longest sequence S^* for $\mathbf{v}$ is calculated. It holds that $S^* = \bar{S}_p(\mathbf{v})$. In the iteration, all speeds of sub-sequence S^* are increased equally. It holds that all customers in S^* have equal speeds in vector $\hat{\mathbf{v}}$ and $\mathbf{v}'$.

Recall, Algorithm 3 increases the speed of customers in S^* until one of the following cases arise:

(1) $t_{l'}' = a_{l'}, u_{l'} \in S^*$,

(2) $\mathbf{v}_k' = \mathbf{v}_j', u_j \in \bar{S}_{p-1}(\mathbf{v})$, or

(3) $\mathbf{v}_k' = v_{max}$.

It follows that $\hat{\mathbf{v}}$ builds a match with S, which is $\tilde{S}(\hat{\mathbf{v}}) = \tilde{S}(\mathbf{v})$ and $\bar{S}_i(\hat{\mathbf{v}}) = \bar{S}_i(\mathbf{v}), i = 1, \ldots, p$ since

- $v_j' = v_j, u_j \in S \setminus S^*$,
- $\hat{v}_j > \hat{v}_k, u_j \in \bar{S}_{p-1}(\hat{\mathbf{v}})$ and $u_k \in \bar{S}_p(\hat{\mathbf{v}})$ and
- $t_j \neq a_j, u_j \in \bar{S}_p(\hat{\mathbf{v}})$, holds.

It remains to show that $\mathbf{v}'$ builds a match with S. We distinguish three cases:

(1) It holds $t_{l'}' = a_{l'}$, for a customer $u_{l'} \in S^*$. It follows that $\mathbf{v}'$ has the following matching partition $\tilde{S}(\mathbf{v}')$ and $\bar{S}(\mathbf{v}') = \bar{S}_1(\mathbf{v}')$, where $u_j \in \tilde{S}(\mathbf{v}')$, for $j \leq l'$, and $u_j \in \bar{S}(\mathbf{v}')$, for $j > l'$.

(2) It holds $v_k' = v_j'$, $u_j \in \bar{S}_{p-1}(\mathbf{v})$, and $t_j' \neq a_j, u_j \in \bar{S}_{p-1}(\mathbf{v}) \cup \bar{S}_p(\mathbf{v})$. It follows that $\mathbf{v}'$ has the following matching partition $\tilde{S}(\mathbf{v}'), \bar{S}_i(\mathbf{v}'), i = 1, \ldots, p-1$, with

- $\tilde{S}(\mathbf{v}') = \tilde{S}(\mathbf{v})$,
- $\bar{S}_i(\mathbf{v}') = \bar{S}_i(\mathbf{v}), i = 1, \ldots, p-2$, and
- $\bar{S}_{p-1}(\mathbf{v}') = \bar{S}_{p-1}(\mathbf{v}) \cup \bar{S}_p(\mathbf{v})$.

(3) It holds $v_k' = v_{max}, t_j' \neq a_j$, for all $u_j \in S^*$.Thus, $p = 1$ and $\mathbf{v}'$ has the following matching partition $\tilde{S}(\mathbf{v}') = \tilde{S}(\mathbf{v})$ and $\bar{S}_1(\mathbf{v}') = \bar{S}_1(\mathbf{v}) \cup \bar{S}_2(\mathbf{v})$. □

Proof of Claim 4 We prove this claim by contradiction. Based on the partitioning, it follows $t_i > a_i$ for $u_i \in \bar{S}(\mathbf{v})$. Let $u_i \in \bar{S}(\mathbf{v})$ denote a customer with $t_i > t_{i-1} + \tau_i + d_i/v_i$. It follows that t_i can be reduced since $t_i > a_i$ holds. This is in contradiction to the assumption that the service start time are early as possible. □

Proof of Lemma 2 We prove this lemma by contradiction. Let $\tilde{\mathbf{v}} = (\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_k)$ denote the first, infeasible speed vector to sequence S, which is calculated by Algorithm 3 and added to $\mathcal{V}$ in some iteration. Let j denote the smallest index with $\tilde{v}_j \notin [v_{min}, v_{max}]$ or $\tilde{t}_j \notin [a_j, b_j]$.

Let $\mathbf{v}^*$ denote the optimal speed vector of the input of Algorithm 3. It holds $\tilde{\mathbf{v}} \neq \mathbf{v}^*$ since $\mathbf{v}^*$ is optimal and thus, feasible.

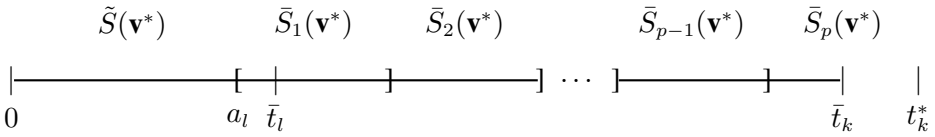

Figure 9 Service Start Times Based on the Speed Vector $\bar{\mathbf{v}}$.

In the following, we focus on the iteration, where $\tilde{\mathbf{v}}$ is added to $\mathcal{V}$. Let $\mathbf{v} = (v_1, v_2, \ldots, v_k)$ denote the speed vector at the beginning of this iteration and let $\mathbf{v}' = (v'_1, v'_2, \ldots, v'_k)$ denote the speed vector at the end of this period. Note that $\mathbf{v} = (v_1, v_2, \ldots, v_k)$ is feasible, otherwise $\tilde{\mathbf{v}}$ is not the first infeasible speed vector. Based on Claim 3, a match exists for speed vector $\mathbf{v}$. Let $\tilde{S}(\mathbf{v}), \bar{S}_i(\mathbf{v}), i = 1, \ldots, p$, denote the matching partitioning of S with vector $\mathbf{v}$.

It holds that $u_j \in \bar{S}_p(\mathbf{v})$ since only speeds in $\bar{S}_p(\mathbf{v})$ are adjusted and speed changes in $\bar{S}_p(\mathbf{v})$ does not impact the rest of the sequence. From here, we distinguish four cases: (1) $\tilde{v}_j > v_{max}$, (2) $\tilde{v}_j < v_{min}$, (3) $\tilde{t}_j > b_j$, or (4) $\tilde{t}_j < a_j$.

(1) It holds $v_j < v_{max}$ since $v'_j \leq v_{max}$ (see line 16) and $\tilde{v}_j \in (v_j, v'_j]$ (see line 23). Thus, $\tilde{v}_j \leq v_{max}$ holds.

(2) First, we show that $v'_j > v_j$ holds. It holds

- $v_j < v_{max}$,
- $v_i > v_j$, $u_i \in \bar{S}_{p-1}(\mathbf{v}), u_j \in \bar{S}_p(\mathbf{v})$, if $p > 1$, and
- $t_i > a_i$, for $u_i \in \bar{S}_p(\mathbf{v})$.

We distinguish the three following cases: (2.1) $v'_j = v_{max}$, (2.2) $v'_j = v_i, u_i \in \bar{S}_{p-1}(\mathbf{v})$, or (2.3) based on $\mathbf{v}'$ it holds $t'_i = a_i$, with $u_i \in \bar{S}_p(\mathbf{v})$. In cases (2.1) and (2.2), it directly follows $v'_j > v_j$. In case (2.3), it holds $a_i = t'_i < t_i$. It holds that the speeds in $\mathbf{v}$ for sub-sequence $\bar{S}_p(\mathbf{v})$ are equal. Same holds for speed vector $\mathbf{v}'$. Thus, $v'_j > v_j$ follows. Based on this conclusion, we can follow that $v_{min} < v_j < \tilde{v}_j \leq v'_j$ holds.

(3) Based on case (2), it follows $v_i < \tilde{v}_i \leq v'_i, u_i \in \bar{S}_p(\mathbf{v})$. Thus, $t'_j \leq \tilde{t}_j < t_j \leq b_j$ holds.

(4) Let $u_{l'}$ denote the last customer in $\bar{S}_{p-1}(\mathbf{v})$ or $\tilde{S}(\mathbf{v})$ and let $t'_{l'}$ denote the service start time based on $\mathbf{v}'$. If no such customer exist, choose $t'_{l'} = 0$. It follows $\tilde{v}_j \leq \sum_{i=l'+1}^{j} d_i/(a_j - t'_{l'})$ (see line 16). It follows

$$a_j > t'_{l'} + \frac{\sum_{i=l'+1}^{j} d_i}{\tilde{v}_i} \geq t'_{l'} + \frac{\sum_{i=l'+1}^{j} d_i}{\sum_{i=l'+1}^{j} d_i}(a_j - t'_{l'}) = a_j,$$

which is a contradiction. This concludes case (4) and it follows that $\tilde{\mathbf{v}}$ has to be feasible. □

Proof of Proposition 7 We prove this proposition by contradiction. Assume there is no cost-minimal speed vector $\bar{\mathbf{v}}$ with $\bar{t}_k = t'_k$ and $\bar{v}_i = v^*_i$ for all customers $u_i \in \tilde{S}(\mathbf{v}^*)$. Let u_l denote the last customer of $\tilde{S}(\mathbf{v}^*)$. Based on the division of S, it holds $t^*_l = a_l$. Thus, $\bar{t}_l > a_l$ for speed vector $\bar{\mathbf{v}}$, otherwise $\mathbf{v}^*$ is already an cost-minimal speed vector for sub-sequence $\tilde{S}(\mathbf{v}^*)$. A visualisation of the service start times with $\bar{\mathbf{v}}$ is given in Figure 9. Note that the speeds of all customers in $\bar{S}_1(\mathbf{v}^*)$ in vector $\mathbf{v}^*$ are equal. We distinguish the two following cases

(1) $\bar{v}_l \leq v^*_l$ or

(2) $\bar{v}_l > v^*_l$.

(1) Based on Claim 2, it holds that $v^*_l \leq v^*_{l+1}$ since $t^*_l = a_l$ and $a_l \neq b_l$. It follows that $\bar{S}_1(\mathbf{v}^*)$ contains a smallest index i with $i > l$ and $\bar{v}_i < \bar{v}_{i+1}$. Such an index needs to be contained since $\bar{t}_l > t^*_l = a_l$, $\bar{v}_l \leq v^*_l$, and either the last customer of $\bar{S}_1(\mathbf{v}^*)$ is served on their upper time window bound with vector $\mathbf{v}^*$ or $p = 1$ and $\bar{t}_k < t^*_k$ holds. Based on Proposition 3, it follows $\bar{t}_i = a_i$. However, this is a contradiction to the feasibility of $\mathbf{v}^*$ since based on Claim 4, $\mathbf{v}^*$ does not result in idle times in $\bar{S}(\mathbf{v}^*)$, $t^*_l < \bar{t}_l$, and $v^*_j \geq \bar{v}_j$ for $j = l, \ldots, i$.

(2) Let index i denote the greatest index with $i < l$ and $\bar{v}_i \leq v_i^*$. Such an index needs to be contained since $\bar{v}_l > v_l^*$ and $\bar{t}_l > t_l^*$ holds. It holds that $\bar{t}_j > t_j^*$ for $i \leq j \leq l$. We distinguish two cases either (2.1) $v_i^* > v_{i+1}^*$ or (2.2) $\bar{v}_i < \bar{v}_{i+1}$. However, both cases lead to contradictions. In case (2.1), it follows that $t_i^* = b_i$, which is a contradiction to the assumption that $\bar{\mathbf{v}}$ has minimal cost. In case (2.2), it follows that $\bar{t}_i = a_i$, which is a contradiction to the feasibility of $\mathbf{v}^*$. Thus, vector $\bar{\mathbf{v}}$ is not cost-minimal. □

Proof of Proposition 8 We prove this proposition by contradiction. Let $\mathbf{v} = (v_1, v_2, \ldots, v_k)$ denote the first speed vector in $\mathcal{V}$, which is not of minimal cost. Let t_k denote the service start time at u_k based on $\mathbf{v}$. This means that a speed vector $\hat{\mathbf{v}} = (\hat{v}_1, \hat{v}_2, \ldots, \hat{v}_k)$ exists such that $t_k = \hat{t}_k$ and the cost of $\hat{\mathbf{v}}$ are less compared to $\mathbf{v}$.

Let $\mathbf{v}^*$ denote the speed vector of the beginning of the iteration, where $\mathbf{v}$ is calculated. Note that $\mathbf{v}^*$ has minimal-cost for the corresponding service time at u_k. This argument holds true since the input speed vector of Algorithm 3 has minimal cost.

Based on Claim 3, there exists a matching partitioning for $\mathbf{v}^*$, denoted by $\tilde{S}(\mathbf{v}^*)$, $\bar{S}_i(\mathbf{v}^*)$, $i = 1, \ldots, p$. Note that $\mathbf{v}$ and $\mathbf{v}^*$ only differ for customers in $\bar{S}_p(\mathbf{v}^*)$.

Let $u_{l'}$ denote the last customer of $\bar{S}_{p-1}(\mathbf{v}^*)$. Based on Proposition 2, it follows that $t_{l'}^* = b_{l'}$. Consider the first customer, where $\mathbf{v}$ and $\hat{\mathbf{v}}$ differ. If this first difference is for a customer $u_j \in \bar{S}(\mathbf{v}^*)_i, i = 1, \ldots, p-1$, then it follows that $\hat{t}_{l'} \neq b_{l'}$ since the speed vector $\mathbf{v}^*$ of the beginning of the iteration is cost-minimal for the sub-sequence $(u_1, \ldots, u_{l'})$.

We distinguish between two cases:

(1) at least two consecutive customers contained in sub-sequence $\bar{S}_p(\mathbf{v}^*)$ have a different speed in $\hat{\mathbf{v}}$ and $v_i = \hat{v}_i$ for all $u_i \in S \setminus \bar{S}_p(\mathbf{v}^*)$ or

(2) a customer contained in a sub-sequence $\bar{S}_\ell(\mathbf{v}^*)$, with $\ell = 1, \ldots, p-1$, is not served at their upper time window bound.

Note that based on Proposition 7, we do not need to consider customers in $\tilde{S}(\mathbf{v}^*)$.

(1) Let u_l denote the last customer of $\bar{S}_{p-1}(\mathbf{v}^*)$, or if $p = 1$, it follows that u_l is the last customer of $\tilde{S}(\mathbf{v}^*)$. It holds that $t_l = \hat{t}_l$, $t_k = \hat{t}_k$, and $v_{l+1} = \cdots = v_k$. Based on Lemma 2, vector $\mathbf{v}$ is feasible and Proposition 1 shows that equal speeds minimise the energy required. Moreover, Algorithm 2 lifts the speeds to the best feasible speed. Thus, $\hat{\mathbf{v}}$ can not be cost-minimal.

(2) Select ℓ minimal. Let u_l be the last customer of either $\bar{S}_{\ell-1}(\mathbf{v}^*)$, $\tilde{S}(\mathbf{v}^*)$, or u_l does not exist. Let $u_{l'}$ be the last customer of $\bar{S}_\ell(\mathbf{v}^*)$. It holds that $\hat{t}_l = t_l$ or both are 0. We distinguish between three cases:

(2.1) $\hat{v}_{l+1} = \cdots = \hat{v}_{l'} > v_{l+1} = \cdots = v_{l'}$,

(2.2) there are two customers u_j, u_{j+1} in $\bar{S}_\ell(\mathbf{v}^*)$ with $\hat{v}_j > \hat{v}_{j+1}$ and $\hat{v}_{j+1} = \cdots = \hat{v}_{l'}$ holds, or

(2.3) there are two customers u_j, u_{j+1} in $\bar{S}_\ell(\mathbf{v}^*)$ with $\hat{v}_j < \hat{v}_{j+1}$ and $\hat{v}_{j+1} = \cdots = \hat{v}_{l'}$ holds.

A visualisation of case (2.1) is given in Figure 10. It holds that $\hat{t}_l = t_l$. It follows $\hat{v}_{l'} \leq \hat{v}_{l'+1}$, otherwise Proposition 2 leads to a contradiction since $\hat{t}_{l'} \neq b_{l'}$. Let u_j denote the first customer with $\hat{v}_j > \hat{v}_{j+1}$ and $l' < j$. Such a customer needs to be contained; otherwise it holds that $\hat{t}_k < t_k$. Based on Proposition 2, it follows $\hat{t}_{j'} = b_{j'}$. However, this contradicts the assumption that $\mathbf{v}$ is feasible since $v_{l+1} = \cdots = v_{l'} > v_{l'+1} \geq \cdots \geq v_k$ holds. Based on Claim 4, $\bar{S}(\mathbf{v})$ does not result in idle times based, and Lemma 2 shows that $\mathbf{v}$ is feasible.

(2.2) A visualisation of this case is given in Figure 11. Since index j is maximal, with $l < j < l'$, it follows that $\hat{v}_{j+1} = \hat{v}_{j+2} = \cdots = \hat{v}_{l'}$. Based on Proposition 2 it follows $\hat{t}_j = b_j$. However, this contradicts the assumption that $\mathbf{v}$

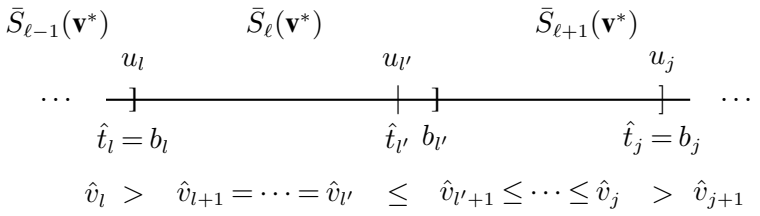

Figure 10 Visualisation of Case (2.1).

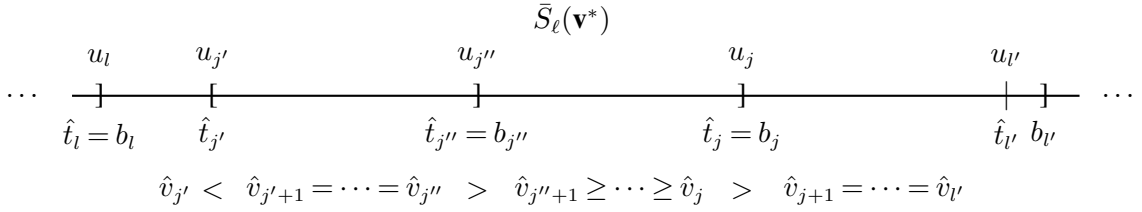

Figure 11 Visualisation of Case (2.2).

has equal speeds for all customers contained in $\bar{S}_\ell(\mathbf{v}^*)$ since $\mathbf{v}$ would violate the time window of customer u_j. Thus, it exists a customer $u_{j'}$ in $\bar{S}_\ell(\mathbf{v}^*)$ with $\hat{v}_{j'} < \hat{v}_{j'+1}$. Choose j' with greatest index with $j' < j$. Let $u_{j''}$ denote the last customer for which $\hat{v}_{j'+1} = \hat{v}_{j'+2} = \cdots = \hat{v}_{j''}$ holds, with $j' < j'' \leq j$. Based on Proposition 2, it holds that $\hat{t}_{j''} = b_{j''}$. However, this also contradicts the assumption that S is feasible with $\mathbf{v}$ since customers $u_{j'+1}, \ldots, u_{j''}$ require an equal speed that is at least $\hat{v}_{j''}$. Thus, this speed is a lower bound for all customers in $\bar{S}_\ell(\mathbf{v}^*)$ if all speeds should be equal. However, with speed $\hat{v}_{j''}$ the service start time at customer $u_{l'}$ is smaller than $t_{l'} < b_{l'}$, which is a contradiction.

(2.3) Since index j is maximal, with $l < j < l'$, it follows $\hat{v}_{j+1} = \cdots = \hat{v}_{l'}$. Based on Proposition 3, it follows $\hat{t}_j = a_j$. From here, we distinguish between the two following cases:

(2.3.1) $\hat{v}_{l+1} \leq \cdots \leq \hat{v}_j$ or

(2.3.2) it exists a customer $u_{j'}$ with $\hat{v}_{j'} > \hat{v}_{j'+1}$ and $l < j' < j$.

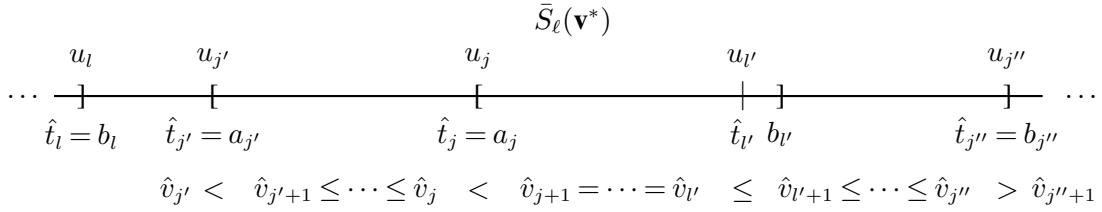

Figure 12 Visualisation of Case (2.3.1).

(2.3.1) Let $u_{j'}$ be the first customer with $l+1 \leq j' < j$ and $\hat{v}_{j'} < \hat{v}_{j'+1}$. It follows

$$v_{l+1} = \cdots = v_{l'} < \hat{v}_{l+1} = \cdots = \hat{v}_{j'} < \hat{v}_{j+1} = \cdots = \hat{v}_{l'}$$

since based on the division of S into sub-sequence it holds $t_j > a_j$. This case is visualised in Figure 12 and is similar to case (2.1). Let $u_{j''}$ denote the first customer with $v_{j''} > v_{j''+1}$ and $l' < j''$. Such a customer has to exist otherwise $\hat{t}_k < t_k$ holds. Based on Proposition 2 it follows $\hat{t}_{j''} = b_{j''}$. However, this contradicts the assumption that $\mathbf{v}$ is feasible since $v_{l+1} = \cdots = v_{l'} > v_{l'+1} \geq \cdots \geq v_k$ holds.

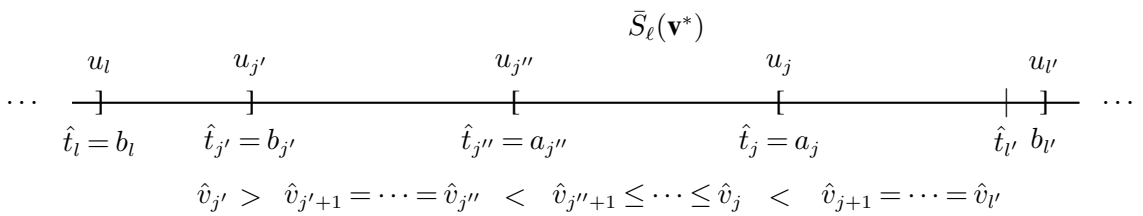

Figure 13 Visualisation of Case (2.3.2).

(2.3.2) Let $u_{j'}$ denote the customer with greatest index such that $\hat{v}_{j'} > \hat{v}_{j'+1}$ and $j' < j$ holds. A visualisation of this case is given in Figure 13. Based on Proposition 2 it holds that $\hat{t}_{j'} = b_{j'}$. Let $u_{j''}$ denote the customer with smallest index such that $j' < j'' \leq j$ and $\hat{v}_{j''} < \hat{v}_{j''+1}$. Based on Proposition 3 it holds that $\hat{t}_{j''} = a_{j''}$. It holds that $\hat{v}_{j'+1} = \cdots = \hat{v}_{j''}$. Moreover, it holds that $\hat{v}_{j'+1} = \cdots = \hat{v}_{j''} < \hat{v}_{j+1} = \cdots = \hat{v}_{l'}$. However, it is impossible to apply an equal speed for all customers of sub-sequence $\bar{S}_\ell(\mathbf{v}^*)$ since customers $u_{j'+1}$ to $u_{j''}$ require a speed that is smaller than $\hat{v}_{j'+1}$ and customers u_{j+1} to $u_{l'}$ need a speed that is at least $\hat{v}_{j+1}$. Thus, vector $\mathbf{v}$ is infeasible, which is a contradiction to Lemma 2. This concludes the proof. □

Proof of Proposition 9 Function ser maps the speeds of customer u_k to their start of the service, which is continuous; see line 21 of Algorithm 3. We prove this proposition by contradiction and assume that a speed vector exists such that u_k is served at t'_k, with $t'_k \neq ser(v_k)$, for all $\mathbf{v} = (v_1, \ldots, v_k)$, $\mathbf{v} \in \mathcal{V}$. Let $\tilde{v}_k$ denote the maximal speed towards customer u_k calculated by Algorithm 3. It follows that $t'_k < ser(\tilde{v}_k)$ since function ser is continuous. Consider, now, the two cases:

(1) $\tilde{v}_k < v_{max}$. In this case, it follows that $ser(\tilde{v}_k) = a_k$. Otherwise, the algorithm increases the speed of u_k further, which leads to a contradiction.

(2) $\tilde{v}_k = v_{max}$. Consider the last iteration of Algorithm 3, where v_k is increased up to v_{max}. Let $\mathbf{v}$ denote the speed vector at the beginning of the last iteration. Based on Claim 3, there is a match for $\mathbf{v}$. Let $\tilde{S}(\mathbf{v}), \bar{S}_i(\mathbf{v}), i = 1, \ldots, p$, denote this partitioning. At the end of the last iteration, all customers in $\bar{S}_p(\mathbf{v})$ have a speed of v_{max}, which means that the time to perform $\bar{S}_p(\mathbf{v})$ cannot be reduced further. Thus, it has to hold that the time to perform a preceding sub-sequence $\bar{S}_i(\mathbf{v})$, with $i = 1, \ldots, p-1$, can be reduced further. Changing speeds of customers contained in $\tilde{S}(\mathbf{v})$ cannot reduce the service start time at u_k since the last customer of $\tilde{S}(\mathbf{v})$ is served on their lower time window bound. Based on Definition 1,the speed is decreasing for two consecutive sub-sequences $\bar{S}_i(\mathbf{v})$ and $\bar{S}_{i+1}(\mathbf{v})$. Thus, it follows that $\bar{S}_p(\mathbf{v})$ can only preceded by a single sub-sequence $\bar{S}_1(\mathbf{v})$ with an equal speed of v_{max}; otherwise the algorithm would need an additional iteration. However, if $\bar{S}_p(\mathbf{v})$ is only preceded by a single sub-sequence $\bar{S}_1(\mathbf{v})$ with a speed of v_{max}, the service start time t'_k cannot exist. □

Proof of Proposition 10 Let k denote the number of customers in S. In the worst case, each customer in sequence S has a different speed, and these speeds are strongly monotonously increasing considered from back to front. This means that the speed of the last customer is increased until it is equal to that of the penultimate customer. Next, the speeds of both are increased until equal to that of the preceding customer. This process repeats until all customers have the same speed. Then, the speed of all customers is increased such that the first customer in S has their service starting at the lower time window bound. Next, the speed of the entire sequence S without the first customer is increased until the second customer in S is served at their lower time window bound. This process repeats until the service start time

of the last customer is at their lower time window bound, and the algorithm terminates. This leads to $2k-1$ many speed increases. In each iteration, the sub-sequence and the increased speed have to be calculated, requiring at most $2k$ calculations. Each customer can only once enter the set of customers that have the same speed of the last.

Proof of Theorem 4 Based on Lemma 2, each speed vector contained in set $\mathcal{V}$ is feasible. The cost function $cost$ is correct since all speed vectors are of minimal cost; see Proposition 8, and the value of the cost function is a simple summation of distances multiplied by the value of function f depending on the speed; see line 22 of Algorithm 2. The service start time function ser is correct since Proposition 9 states that Algorithm 3 calculates all possible service start times at the last customer of a sequence. Thus, the correctness of Algorithm 3 is proven. The polynomial running time is shown in Proposition 10. □

Proof of Proposition 11 Let $L^A = (u_k, W^A, \mathbf{v}^A, q^A, \bar{c}^A, \mathcal{V}^A, ser^A, cost^A)$ and $L^B = (u_k, W^B, \mathbf{v}^B, q^B, \bar{c}^B, \mathcal{V}^B, ser^B, cost^B)$ denote two labels. Suppose L^A fulfils Definition 2 but does not dominate L^B. It follows that there is a partial path $\tilde{W}$ that extends label L^B to the sink node. From here, we distinguish the two following cases:

(1) label L^A can not be extended via $\tilde{W}$ or

(2) the reduced cost of label L^A extended via $\tilde{W}$ is greater than the reduced cost of the extension of label L^B.

(1) This case can be divided further into:

(1.1) $(k,i) \notin \bar{A}$ for any arc in $\tilde{W}$,

(1.2) $\tilde{W}$ contains a node for which $u_i \in X^A$ holds,

(1.3) the extension of L^A via $\tilde{W}$ is in conflict with the load constraint, or

(1.4) there is no feasible speed vector for L^A extended via $\tilde{W}$.

Case (1.1): This leads to a contradiction as the extension of L^B via $\tilde{W}$ is feasible.

Case (1.2): By Condition 15, it holds that $Y = \emptyset$. Thus, for any $u_i \in X^A \setminus X^B$, we have

$$a_k + \tau_k + \frac{d_{ki}}{v_{\max}} > b_i,$$

which implies that the earliest possible arrival at u_i from u_k violates the time window of u_i.

Now suppose that u_i is visited at a later position in $\tilde{W}$, i.e., not directly after u_k. Then, the arrival at u_i would only be delayed further due to additional travel and service times. Hence, the time window violation cannot be resolved by repositioning u_i in $\tilde{W}$, and the triangle inequality ensures that detouring cannot reduce the travel time. It follows that any extension of L^B that includes u_i is infeasible.

Case (1.3): By Condition (12), it holds that $q^A \leq q^B$. Thus, the extension of L^B also violates the load constraint.

Case (1.4): Assume that label L^A cannot be extended via $\tilde{W}$ because every such extension violates a time window constraint. Let $\ell^A = |W^A|$ and $\tilde{\ell} = |\tilde{W}|$. Define

$$\mathbf{v}^A = \arg\min_{\mathbf{v} \in \mathcal{V}^A} ser^A(\mathbf{v}_{\ell^A}),$$

the vector in $\mathcal{V}^A$ giving the earliest arrival at u_k. Then form the extended vector

$$\tilde{\mathbf{v}}^A = (v_1^A, \ldots, v_{\ell^A}^A, v_{\max}, \ldots, v_{\max})$$

of total length $\ell^A + \tilde{\ell}$, where the last $\tilde{\ell}$ entries correspond to arcs in $\tilde{W}$.

By assumption no fully feasible extension exists, so $\tilde{\mathbf{v}}^A$ violates at least one time window. Let u_i be the first customer in $\tilde{W}$ based on $\tilde{\mathbf{v}}^A$ for which

$$\tilde{t}_i^A = \tilde{t}_{k'} + \tau_{k'} + \sum_{u_j \in \tilde{W} \setminus u_i} (\frac{d_j}{v_{\max}} + \tau_j) + \frac{d_i}{v_{\max}} > b_i,$$

where

- $u_{k'}$ is the last customer on $\tilde{W}$ at which the vehicle idles before u_i (or u_k if none),
- $\tilde{t}_{k'} = a_{k'}$ or is $ser^A(\mathbf{v}^A_{\ell^A})$ if $k' = k$ holds.

Let $\ell^B = |W^B|$. By Condition (13), the earliest feasible service time at u_k in L^A is less than or equal to that in L^B. Since both vectors use the same maximal speed beyond u_k, the subsequent arrival times for any downstream customer are also earlier (or equal) for L^A. Hence, $t_j^B \geq t_j^A > b_j$, which contradicts the feasibility of L^B. Hence Case (1.4) is impossible. Since all four subcases lead to contradictions, Case (1) cannot occur.

Case (2): Let $\bar{W}^A = W^A + \tilde{W}$ and $\bar{W}^B = W^B + \tilde{W}$ denote two paths, where $\tilde{W}$ are feasible extensions of the labels L^A and L^B to the sink node w_{n+1}. Let

$$\bar{\mathbf{v}}^B = (\bar{v}_1^B, \ldots, \bar{v}_{\ell^B}^B, \bar{v}_{\ell^B+1}^B, \ldots, \bar{v}_{\ell^B+\tilde{\ell}}^B)$$

be an optimal speed vector for $\bar{W}^B$. By Condition (13) and (14), it holds that a vector $\mathbf{v}^A = (v_1^A, \ldots, v_{\ell^A}^A)$ to W^A exists such that the service start time at u_k is not greater than the service start time of u_k with W^B and $(\bar{v}_1^B, \ldots, \bar{v}_{\ell^B}^B)$ and the reduced cost of W^A with $\mathbf{v}^A$ is less than or equal to the reduced cost of W^B with $(\bar{v}_1^B, \ldots, \bar{v}_{\ell^B}^B)$.

We define a candidate speed vector for $\bar{W}^A$ by reusing the suffix of $\bar{\mathbf{v}}^B$ on $\tilde{W}$, i.e.,

$$\bar{\mathbf{v}}^A = (v_1^A, \ldots, v_{\ell^A}^A, \bar{v}_{\ell^A+1}^B, \ldots, \bar{v}_{\ell^A+\tilde{\ell}}^B).$$

This vector is feasible for $\bar{W}^A$ because: (i) the suffix is feasible on $\tilde{W}$, (ii) $\tilde{W}$ is equal for $\bar{W}^A$ and $\bar{W}^B$, and (iii) the service start time at customer u_k is not later for $\bar{\mathbf{v}}^A$ than for $\bar{\mathbf{v}}^B$. Since both $\bar{W}^A$ and $\bar{W}^B$ share the suffix $\tilde{W}$ and this partial path is traversed using the same speed vector, the cost incurred along $\tilde{W}$ is identical for both paths. Thus, any difference in reduced cost must come from the prefix. Hence, the total reduced cost of $\bar{W}^A$ is less than or equal to that of $\bar{W}^B$, contradicting the assumption of Case (2) that the extension of L^A has strictly higher reduced cost. Therefore, Case (2) cannot occur. □

B. Pseudocode of Labelling Algorithm

Algorithm 4 computes the set of all non-dominated labels that represent feasible paths from the depot w_0 to the sink node w_{n+1} in the directed graph $\bar{D} = (\bar{V}, \bar{A})$. In lines 3-4 it initialises all label sets $\mathcal{K}_i$ as empty. Non extended labels are stored in a list $\mathcal{L}$. This list is initialised with a label at the source node w_0. While $\mathcal{L}$ is not empty (line 5), the first label is selected and evaluated for all feasible successor nodes $u_j \in \bar{V}$ (line 7). A successor is considered valid if it has not yet been visited, if the arc (i, j) exists in the graph, if the total demand remains within capacity, and if Algorithm 2 can compute a feasible speed vector for the extended path (line 8). If these conditions are satisfied, a new label L is created by extending the current label to node u_j (line 9). If the new node is the sink node w_{n+1}, the label is added to the final solution set $\mathcal{K}_{n+1}$ (line 11). Otherwise, the algorithm checks whether the new label is dominated or dominates existing labels at node u_j (lines refalgo:labelling:dominance-start–20). Dominated labels are pruned accordingly. Finally, if the new label is non-dominated and $u_j \neq w_{n+1}$, it is added to the label set $\mathcal{K}_j$ and to the processing list $\mathcal{L}$ for further extension (line 22). The process repeats until all extendable labels have been processed.

Algorithm 4 Labelling Algorithm for Resource-Constrained Shortest Path with Speed Optimization

Input: graph $\bar{D} = (\bar{V}, \bar{A})$
Output: set of non-dominated labels $\mathcal{K}_{n+1}$ at the sink node
$\mathcal{K}_i \leftarrow \emptyset$ for all $u_i \in \bar{V}$
$\mathcal{L} \leftarrow (w_0, (w_0), (), 0, \tilde{\pi}, \emptyset, ser, cost)$ ▷ Initialize start label at depot
while $\mathcal{L} \neq \emptyset$ **do**
  $(u_i, W, \mathbf{v}, q, \bar{c}, \mathcal{V}, ser, cost) \leftarrow$ remove first label from $\mathcal{L}$ ▷ Select label for extension
  for $u_j \in \bar{V}$ **do**
    if $u_j \notin W \wedge (i,j) \in \bar{A} \wedge q + q_j \leq Q$
    and Algo. 2 computes speed vector for $W \cup \{u_i\}$ **then** ▷ Check feasibility of extension and compute speed vector
      create new label L ▷ Extend label to u_j
      if $u_j = w_{n+1}$ **then**
        $\mathcal{K}_{n+1} \leftarrow \mathcal{K}_{n+1} \cup \{L\}$ ▷ Store completed path at sink node
      else
        for $\tilde{L} \in \mathcal{K}_j$ **do**
          if $\tilde{L}$ dominates L **then**
            discard L ▷ New label is dominated; prune it
          else if L dominates $\tilde{L}$ **then**
            remove $\tilde{L}$ from $\mathcal{K}_j$ and $\mathcal{L}$ ▷ Remove dominated label
          end if
        end for
      end if
      if $u_j \neq w_{n+1}$ **and** L is not dominated **then**
        add L to $\mathcal{L}$ and $\mathcal{K}_j$ ▷ Add non-dominated label for further extension
      end if
    end if
  end for
end while

C. Effect of the acceleration techniques

Table 6 and Table 7 show the effect of the different acceleration strategies on individual instances, namely the runtimes (in seconds) separately for each instance and for each configuration. The first part of each table presents the baseline runtime with all acceleration techniques active, alongside the effect of deactivating one technique at a time (i.e., no dominance rule, adding all variables, or no heuristic labelling). The second part reports the runtime when no acceleration is used and subsequently compares the effect of activating only one technique at a time (dominance rule only, reduction of added variables only, and heuristic labelling only).

Table 6 Computational Results to Evaluate Benefit of Acceleration Techniques on UK-A Instances with 10 Customers.

Inst.	All Acc. Tech. [s]	No Dom. [s]	Add All Var. [s]	No Rem. of Arcs [s]	No Acc. Tech. [s]	Only Dom. [s]	Only Red. of Num. of Var. Added [s]	Only Rem. of Arcs [s]
A-10-1	1.79	14.09	2.70	2.47	112.18	3.44	37.90	51.79
A-10-2	0.47	6.68	0.75	1.47	140.54	1.97	23.84	29.81
A-10-3	0.25	2.30	0.77	0.56	474.76	4.27	14.07	33.17
A-10-4	0.48	5.13	4.27	1.21	3085.56	8.49	21.15	119.33
A-10-5	2.08	10.50	2.00	3.11	76.26	2.35	34.45	149.96
A-10-6	0.31	5.34	0.97	1.47	1153.86	14.75	17.55	80.03
A-10-7	0.52	16.86	0.45	1.12	tl	1.16	28.42	311.43
A-10-8	0.20	1.36	0.43	0.56	44.91	1.30	4.89	5.12
A-10-9	0.58	15.39	1.21	1.81	tl	11.71	35.86	1389.42
A-10-10	0.72	8.05	1.90	2.04	419.17	2.98	18.15	193.71
A-10-11	0.16	0.86	0.19	0.39	5.47	0.58	2.60	0.78
A-10-12	1.47	5.64	2.49	1.82	1945.29	15.15	31.47	21.35
A-10-13	0.62	19.35	1.81	0.87	691.79	18.54	51.13	25.96
A-10-14	0.75	12.02	0.83	3.15	297.55	4.10	55.24	45.97
A-10-15	1.43	196.45	2.34	3.28	tl	24.88	376.73	480.01
A-10-16	3.74	23.72	3.72	4.88	384.52	8.74	40.29	51.42
A-10-17	1.62	15.05	2.83	3.41	tl	13.79	62.64	312.84
A-10-18	1.94	27.44	2.46	2.77	2412.72	6.93	79.36	83.97
A-10-19	0.67	18.15	1.69	1.16	1210.13	5.30	33.18	31.61
A-10-20	7.58	75.08	7.86	11.93	71.95	25.08	97.85	57.24

Table 7 Computational Results to Evaluate Benefit of Acceleration Techniques on UK-B Instances with 25 Customers and UK-C with 15 Customers.

Inst.	All Acc. Tech. [s]	No Dom. [s]	Add All Var. [s]	No Rem. of Arcs [s]	No Acc. Tech. [s]	Only Dom. [s]	Only Red. of Num. of Var. Added [s]	Only Rem. of Arcs [s]
B-25-1	0.63	50.86	1.17	0.63	tl	2.38	115.51	27.50
B-25-2	2.95	419.03	4.09	2.96	tl	4.78	697.74	375.24
B-25-3	3.19	2171.81	4.45	3.07	tl	8.38	tl	tl
B-25-4	9.75	793.63	6.95	4.16	tl	5.37	952.59	273.67
B-25-5	5.90	722.63	7.58	5.90	tl	6.58	1007.41	tl
B-25-6	2.02	81.15	3.40	0.76	tl	1.98	92.33	57.52
B-25-7	9.72	2340.94	9.94	5.98	tl	20.35	2866.00	1689.54
B-25-8	1.38	53.72	2.16	1.55	1668.80	3.01	102.63	42.84
B-25-9	4.35	907.30	6.63	3.96	tl	7.64	1182.05	1313.83
B-25-10	15.20	878.78	18.17	16.40	tl	21.91	1015.38	836.52
B-25-11	84.20	1491.71	79.56	80.17	tl	80.95	1515.78	1306.42
B-25-12	0.14	2.71	0.23	0.18	124.17	0.29	4.52	2.59
B-25-13	128.36	tl	220.75	150.69	tl	284.40	tl	tl
B-25-14	0.66	11.67	0.69	0.75	185.66	0.93	15.93	9.45
B-25-15	0.74	12.18	0.77	0.59	176.02	1.11	21.04	10.24
B-25-16	0.70	114.16	1.16	1.07	tl	2.00	162.16	105.82
B-25-17	0.49	3.52	0.46	0.40	10.39	0.44	5.57	2.70
B-25-18	0.20	5.09	0.26	0.20	25.04	0.39	10.94	5.30
B-25-19	0.43	15.01	0.49	0.43	618.18	0.72	19.04	10.67
B-25-20	32.66	517.25	45.30	29.31	901.93	30.98	472.33	740.79
C-15-1	2.05	12.12	1.20	1.61	429.18	2.37	26.25	13.22
C-15-2	0.61	25.13	0.55	1.01	2412.74	4.98	70.83	102.77
C-15-3	0.44	2.26	0.46	0.56	21.10	0.59	4.58	3.62
C-15-4	36.05	162.32	28.13	41.19	tl	45.01	257.35	304.73
C-15-5	0.18	1.50	0.45	0.24	16.67	0.38	3.90	5.42
C-15-6	12.14	219.27	8.04	10.51	tl	17.11	279.50	296.46
C-15-7	0.82	9.22	0.68	0.66	26.30	0.82	17.97	17.39
C-15-8	8.44	353.72	8.97	11.78	tl	19.05	805.34	2910.40
C-15-9	20.86	154.56	17.62	24.46	tl	22.53	274.75	307.18
C-15-10	2.09	21.98	2.45	4.88	222.23	25.62	104.43	247.45
C-15-11	0.87	12.52	0.62	0.84	1796.29	3.20	24.07	24.27
C-15-12	0.17	1.44	0.24	0.24	47.40	0.28	3.63	4.20
C-15-13	0.71	20.85	3.30	1.28	tl	9.33	56.34	1496.35
C-15-14	0.28	0.60	0.31	0.33	4.58	0.37	1.52	2.94
C-15-15	0.63	36.49	0.98	1.01	tl	5.99	129.73	1202.74
C-15-16	5.44	96.09	2.57	5.25	tl	7.68	192.86	140.54
C-15-17	0.23	1.45	0.23	0.25	7.29	0.48	3.73	2.56
C-15-18	0.39	2.17	0.67	0.59	75.06	1.11	5.10	3.77
C-15-19	2.35	68.54	2.24	4.41	tl	4.00	190.86	318.47
C-15-20	1.07	36.98	0.60	0.89	tl	1.10	77.68	48.82