Appendix. Algorithm for Section 5.3

Algorithm 1: Short-distance tariff: finding a short-distance path.

Input : PTN (V, E), upper bounds $S_{\max}$, $L_{\max}$, two stations $x, y \in V$
Output: shortest x-y-path W with $s(W) \leq S_{\max}$ and $l(W) \leq L_{\max}$ if one exists

```
// Initialization
```
$S_{\max} := \min\{S_{\max}, |V| - 1\}$, $L_{\max} := \min\{L_{\max}, \max_{e \in E} l(e) \cdot |V|\}$
for all $v \in V$ **do**
    $d_0(v) := \infty$
    $\pi_0(v) :=$ None
$d_0(x) := 0$

```
// Compute distances and predecessors (Bellman-Ford)
```
for $s = 1, \ldots, S_{\max}$ **do**
    for all $v \in V$ **do**
        $d_s(v) := d_{s-1}(v)$
        $\pi_s(v) := \pi_{s-1}(v)$
        for all edges $(w, v) \in E$ **do**
            if $d_s(v) > d_{s-1}(w) + l(w, v)$ **then**
                $d_s(v) := d_{s-1}(w) + l(w, v)$
                $\pi_s(v) := w$

```
// Check if a feasible x-y-path exists and compute it if necessary
```
if $d_{S_{\max}}(y) \leq L_{\max}$ **then**
    `// Determine the path` W `from` x `to` y `by backtracking the predecessors`
    $W' = [y]$ `// list of all predecessors starting from` y
    current $:= y$
    $s := S_{\max}$
    while current $\neq x$ **do**
        current $:= \pi_s$(current)
        W'.append(current) `// add` current `to the end of` W'
        $s := s - 1$
    Set $W := (W')^{-1}$
    return W
else
    return None