Appendix

A. Trans-ULTRA Optimizations

A simplified version of Trans-ULTRA was presented in Section 4.3. In this section, we present additional performance optimizations that are carried over from ULTRA. These are based on the observation that in order to generate a CP transfer set, it is not necessary to find and unpack all canonical journeys, merely those that are candidates.

In the profile search performed from each source stop p_s, this is exploited by skipping all runs that cannot produce any candidates. The set of initial stop events that can occur in a journey with departure time τ_{dep} is given by

$$\mathcal{E}(p_s, \tau_{\text{dep}}) := \{T[i] \mid \tau_{\text{dep}} + \Delta\tau_{\text{fp}}(p_s, p(T[i])) = \tau_{\text{dep}}(T[i])\}.$$

Normally, the profile search performs a run for every departure time τ_{dep} such that $\mathcal{E}(p_s, \tau_{\text{dep}})$ is not empty. We call τ_{dep} a *candidate departure time* if there is a stop event $T[i] \in \mathcal{E}(p_s, \tau_{\text{dep}})$ with $p(T[i]) = p_s$. Let $\tau^1_{\text{dep}}, \ldots, \tau^k_{\text{dep}}$ be the sequence of candidate departure times, sorted in ascending order. To simplify the notation, let $\tau^{k+1}_{\text{dep}} := \infty$. Trans-ULTRA performs a run for each candidate departure time τ^i_{dep} in descending order; the non-candidate departure times are skipped. As a consequence, the run for τ^i_{dep} may find new journeys whose departure time is not exactly τ^i_{dep} but lies in the interval $[\tau^i_{\text{dep}}, \tau^{i+1}_{\text{dep}})$. The proof of Lemma 4 still carries over to a weaker claim, which is sufficient to prove Theorem 2:

LEMMA A.1. *Let J be a candidate that is canonical for the query $Q = (p_s(J), p_t(J), \tau_{\text{dep}}(J))$. After the run for $\tau_{\text{dep}}(J)$ in the canonical rRAPTOR search from $p_s(J)$, the representative for $c(J)$ at $p_t(J)$ is J.*

Another optimization is that the algorithm only maintains parent pointers for representatives whose initial transfer is empty; all other pointer are set to a dummy value. This allows the algorithm to efficiently identify whether a found journey is a candidate and needs to be unpacked. It can also be leveraged by adding an extra ULTRA condition (U3): Let J be the newly found journey and let J' be the previous representative. Then J is discarded if $J' \preceq J$ and J has a non-empty initial transfer. This does not affect the correctness of Lemma A.1 because (U3) is not fulfilled for any prefix of a canonical candidate.