multi-class Traffic Assignment by Paired Alternative Segments (mTAPAS)

$mTAPAS(\epsilon, \theta, tol)$

Input:
ϵ: Minimal flow level
θ: Minimal cost level
tol: Tolerance level

Step 1. Initialize origin-based arc flows x_{ij}^{kr}, origin-based reduced arc cost π_{ij}^{kr}, least cost path predecessor labels for every origin, every vehicle class L_r^k, and an empty set e for paired alternative segments

$x_{ij}^{kr} \to 0 \quad \forall\, k \in K, r \in R, i \in N, j \in H(i)$
$\pi_{ij}^{kr} \to 0 \quad \forall\, k \in K, r \in R, i \in N, j \in H(i)$
$L_r^k \to \{\text{if } i = r\ r \text{ else } -1;\ i \in N\} \quad \forall\, k \in K, r \in R$
$e \to \{\quad\}$

Step 2. Perform All-or-Nothing (AON) assignment – From each origin r, find the least cost path to every destination s, for every vehicle class k, and assign demand q_{rs}^k to this path. Update x_{ij}^{kr} and π_{ij}^{kr} for arcs on this path.

```
for r ∈ R
    for s ∈ S_r
        for k ∈ K
            c_k → {c_ij^k(∑_{r∈R} ∑_{k∈K} x_ij^kr); i ∈ N, j ∈ H(i)}
            L_r^k → label(c_k, r)
            p_rs^k → path(L_r^k, r, s)
            for (i, j) ∈ p_rs^k  x_ij^kr → x_ij^kr + q_rs^k  end
        end
    end
end
```

Step 3. Iterate to shift flows between arcs until the algorithm converges

```
converged → false
while !converged
    for r ∈ R
        for k ∈ K
            c_k → {c_ij^k(∑_{r∈R} ∑_{k∈K} x_ij^kr); i ∈ N, j ∈ H(i)}
            L_r^k → label(c_k, r)
            T_r^k → tree(L_r^k, r)
            Step 3.1 Identify arcs with substantial flow x_ij^kr and substantial reduced cost π_ij^kr
            for i ∈ N
                p_ri → path(L_r^k, r, i)
                u_ri^k → ∑_{(t,h)∈ p_ri} c_th^kr(∑_{r∈R} ∑_{k∈K} x_th^kr)
                for j ∈ H(i)
                    p_rj → path(L_r^k, r, j)
                    u_rj^k → ∑_{(t,h)∈ p_rj} c_th^kr(∑_{r∈R} ∑_{k∈K} x_th^kr)
                    π_ij^kr → u_ri^k + c_ij^kr(∑_{r∈R} ∑_{k∈K} x_ij^kr) − u_rj^k
                    If j ∈ T_r^k and π_ij^kr > θ and x_ij^kr > ϵ
                        Step 3.2. Develop Paired Alternative Segment (PAS) for the potential arc
                        using Maximum Cost Search (MCS) algorithm and perform flow shift
                        (e_1, e_2) → mcs((i, j), k, r)
                        shift((e_1, e_2), k, r)
                        if (e_1, e_2) ≠ ({ }, { }) e → e ∪ ((e_1, e_2), k, r) end
                    end
                end
            end
        end
```

Step 3.3. Randomly sample a subset of PAS from set ρ and perform flow shift to fasten algorithm convergence.

for $\big((e_1, e_2), r\big) \in sample(e)$ $shift\big((e_1, e_2), k, r\big)$ end

end

Step 4. Remove PAS which no longer results in significant improvement in the solution

for $n \in 1{:}\,20$

for $\big((e_1, e_2), k, r\big) \in e$

$c_1 \rightarrow \sum_{(i,j)\in e_1} c_{ij}^{k}\big(\sum_{r\in R}\sum_{k\in K} x_{ij}^{kr}\big)$

$c_2 \rightarrow \sum_{(i,j)\in e_2} c_{ij}^{k}\big(\sum_{r\in R}\sum_{k\in K} x_{ij}^{kr}\big)$

$f_1 \rightarrow \min\ \{x_{ij}^{kr}; (i,j) \in e_1\}$

$f_2 \rightarrow \min\ \{x_{ij}^{kr}; (i,j) \in e_2\}$

if $(f_1 < \epsilon$ or $f_2 < \epsilon)$ and $|c_1 - c_2| > \theta$

$e \rightarrow e \backslash \big((e_1, e_2), k, r\big)$

else

$shift\big((e_1, e_2), k, r\big)$

end

end

end

Step 5. If the relative gap is smaller than the tolerance level then the algorithm is said to have converged

$c_k \rightarrow \{c_{ij}^{k}\big(\sum_{r\in R}\sum_{k\in K} x_{ij}^{kr}\big);\ i \in N, j \in H(i)\}\quad \forall\, k \in K$

$L_r^k \rightarrow label(c_k, r)\quad \forall\, k \in K, r \in R$

$p_{rs}^k \rightarrow path(L_{rs}^k, r, s)\quad \forall\, k \in K, r \in R, s \in S_r$

$u_{rs}^k \rightarrow \sum_{(i,j)\,\in\, p_{rs}^k} c_{ij}^{k}\ \big(\sum_{r\in R}\sum_{k\in K} x_{ij}^{kr}\big)$

$$relative\ gap \rightarrow \frac{\sum_{r\in R}\sum_{s\in S_r}\sum_{k\in K} q_{rs}^k . u_{rs}^k}{\sum_{r\in R}\sum_{k\in K}\sum_{(i,j)\in A} x_{ij}^{kr} . c_{ij}^{k}\big(\sum_{r\in R}\sum_{k\in K} x_{ij}^{kr}\big)}$$

if $\log(relative\ gap) \leq tol$ $converged \rightarrow true$ end

end

return $\{x_{ij}^{kr};\ k \in K, r \in R, i \in N, j \in H(i)\}$

Maximum Cost Search (MCS) algorithm

$mcs(a, k, r)$

Inputs:
a: Arc a as $(i, j); i \in N, j \in H(i)$
r: origin node

$(i, j) \rightarrow a$

Step 1. Initialization –
Step 1.1. Initialize status label l_u for each node. Set l_u to 1 for node i, -1 for all nodes on least cost path between origin node r and node j, and 0 for all other nodes.
$p_{rj} \rightarrow path(L_r^k, r, j)$
$l_u \rightarrow 0 \quad \forall\, u \in N \backslash \{a, p_{rj}\}$
$l_u \rightarrow 1 \quad \forall\, u \in a$
$l_u \rightarrow -1 \quad \forall\, u \in p_{rj}$
Step 1.2. Initialize predecessor label L_u for each node and set L_j to node i.
$L_u \rightarrow -1 \quad \forall\, u \in N \backslash j$
$L_j \rightarrow i$
Step 1.3. Set the tail and head node on arc a.
$t, h \rightarrow i, j$

Step 2. Iterate
while $true$
    Step 2.1. Set the current node to the tail node, and set the tail node to the tail node of the arc with maximum cost headed at the current node
    $v \rightarrow t$
    $t \rightarrow \underset{u \in T(h)}{\text{argmax}}\, c_{hu}^k \left(\sum_{r \in R} \sum_{k \in K} x_{hu}^{kr}\right)$
    Step 2.2. Set the predecessor label of the current node to this tail node.
    $L_v \rightarrow t$
    Step 2.3. If the tail node happens to be on the least cost path between origin node r and node j, i.e., if its status label is -1, then the algorithm can establish a PAS.
    if $l_t = -1$
        Step 2.3.1. Establish the segment between the tail node and node j on the least cost path between origin origin node r and node j as the first segment of PAS – e_1.
        $e_1 \rightarrow path(L_r^k, t, j)$
        Step 2.3.2. Establish the second segment of the PAS – e_2, using predecessor labels backtracking from node j to the tail node. Go to step 3.
        $e_2 \rightarrow path(\{L_u; u \in N\}, t, j)$
        break
    Step 2.4. If the tail node is a previously identified predecessor, i.e., if its status label is 1, then the algorithm has found a cycle. Perform shift flow on this cycle and restart the search process from Step 1
    elseif $l_t = 1$
        $p \rightarrow path(L_u, h, t)$
        $shift(p, \{\quad\}), k, r)$
        break
    Step 2.5. Else update the status of this predecessor and continue to step 2.1
    else
        $l_t = 1$
    end
end

return (e_1, e_2)

Newton Flow Shift (NFS) Method inherited from Dial (2006)

$shift\big((e_1, e_2), k, r\big)$

Inputs:
(e_1, e_2): Paired alternative segments (PAS)
k: Vehicle class associated with PAS
r: Origin associated with PAS

Step 1. Set c_1 and c_2 as the sum of arc costs for vehicle class k on e_1 and e_2 respectively
$c_1 \to \sum_{(i,j)\in e_1} c_{ij}^{k}\left(\sum_{r\in R}\sum_{k\in K} x_{ij}^{kr}\right)$
$c_2 \to \sum_{(i,j)\in e_2} c_{ij}^{k}\left(\sum_{r\in R}\sum_{k\in K} x_{ij}^{kr}\right)$

Step 2. Set c_1' and c_2' as the sum of derivative* of arc cost for vehicle class k on e_1 and e_2 respectively.
$c_1' \to \sum_{(i,j)\in e_1} {c'}_{ij}^{k}\left(\sum_{r\in R}\sum_{k\in K} x_{ij}^{kr}\right)$
$c_2' \to \sum_{(i,j)\in e_2} {c'}_{ij}^{k}\left(\sum_{r\in R}\sum_{k\in K} x_{ij}^{kr}\right)$

Step 3. Set f_1 and f_2 as the minimum arc flow for vehicle class k from origin r on e_1 and e_2 respectively.
$f_1 \to \min\ \left\{x_{ij}^{kr}; (i,j) \in e_1\right\}$
$f_2 \to \min\ \left\{x_{ij}^{kr}; (i,j) \in e_2\right\}$

Step 4. Compute Δ
$\Delta = \frac{c_2 - c_1}{c_1' + c_2'}$

Step 5. Compute δ
if $c_1' + c_2' = 0$ $\delta \to 0$
elseif $\Delta \geq 0$ $\delta \to \min(f_2, \Delta)$
else $\delta \to \max\,(-f_1, \Delta)$

Step 6. Update flow for arcs on e_1 and e_2, for vehicle class k originating from node r flow δ for e_2.
for $(i,j) \in e_1$ $x_{ij}^{kr} \to x_{ij}^{kr} + \delta$ end
for $(i,j) \in e_2$ $x_{ij}^{kr} \to x_{ij}^{kr} - \delta$ end

Dijkstra's labeling algorithm

$label(r,s)$

Input:
r: Origin
s: Destination

Step 1. Initialize a set of open nodes X, cost label C_k and predecessor label L_k for each node
$X \to \{k;\ k \in N\}$
$C_k \to \infty \quad \forall\, k \in N$
$L_k \to 0 \quad \forall\, k \in N$

Step 2. Develop Dijkstra's labels
Step 2.1. Set origin as the pivot node, origin cost label to zero, origin predecessor label to itself, and remove the pivot node from the set of open nodes
$i \to r$
$C_i \to 0$
$P_i \to i$
$X \to X \backslash i$
while $X \neq \emptyset$
    Step 2.2 Update cost and predecessor label for every neighboring node from the pivot node
    for $j \in H(i)$
        if $C_j > C_i + c_{ij}$
            $C_j \to C_i + c_{ij}$
            $L_j \to i$
        end
    end
    Step 2.3. From the set of open nodes, set the node with the smallest cost label as the pivot node, and remove it from the set of open nodes
    $i \to \underset{k\,\in\,X}{\operatorname{argmin}} \{C_k; k \in X\}$
    $X \to X \backslash i$
end

return L

Developing least cost path from Dijkstra's labels

$path(L, r, s)$

Input:
L: Predecessor labels $\{L_i; i \in N\}$
r: Origin
s: Destination

Step 1. Start from destination and visit predecessors using the predecessor labels until arriving at the origin
$p \to \{\ \ \}$
$i \to s$
while $i \neq r$
    $p \to p \cup \{(L_i, i)\}$
    $i \to L_i$
end

Step 2. Revert the sequence of nodes followed from destination to origin
reverse p

return p

Developing least cost tree from Dijkstra's labels

$tree(L, r)$

Input:
L: Predecessor labels $\{L_i; i \in N\}$
r: Origin

Step 1. Initialize an empty set T
$T \to \{\ \ \}$

Step 2. Connect every node to its predecessor node using the predecessor label to develop the least cost tree rooted at origin node r
for $i \in N$
    if $i \neq L_i$ and $i \neq -1$
        $T \to T \cup \{(L_i, i)\}$
    end
end

return T