Master thesis

Omnitrans International
Universiteit Twente

Static user equilibrium
a search for an optimal algorithm

S.W. Ton
0131385

Stochastic Operations Research,
Applied Mathematics.
Supervisors: W.R.W. Scheinhardt,
M.P. Schilpzand.

UNIVERSITEIT TWENTE.

OmniTRANS
Transport Planning Software

Summary

There are many algorithms which converge to a static user equilibrium (UE) in a network without junction modeling. We have investigated eight algorithms and compared these algorithms based on convergence rate, memory usage and proportionality. We considered "Traffic Assignment by Paired Alternative Segments" (TAPAS) as a promising algorithm. We have found modifications for TAPAS such that the convergence rate may increase and we investigated a manner in which we can modify it in such a way that it is also useful in networks with junction modelling. We have included a proof of the convergence of TAPAS, the existence of a unique UE in terms of link-flow in networks without junction modelling and, under certain conditions, in networks with junction modelling.

Samenvatting

Er zijn veel algoritmen die convergeren naar een gebruikersevenwicht in een netwerk zonder kruispuntmodellering. We hebben acht algoritmen onderzocht en vergeleken op basis van convergentiesnelheid, geheugenverbruik en proportionaliteit. Wij vinden "Traffic Assignment by Paired Alternative Segments" (TAPAS) een veelbelovend algoritme. We hebben modificaties voor TAPAS gevonden zodanig dat de convergentiesnelheid wellicht kan worden verbeterd en we hebben een manier gezocht om kruispuntmodellering mogelijk te maken bij TAPAS. We hebben ook een bewijs van convergentie van TAPAS, het bestaan van een uniek gebruikersevenwicht in termen van link-flow in netwerken zonder kruispuntmodellering en, onder bepaalde voorwaarden, in netwerken met kruispuntmodellering toegevoegd.

Preface

This thesis is the final work of my graduation study at Stochastic Operations Research (SOR), department of Applied Mathematics (AM), University of Twente. The corresponding research has been conducted at Omnitrans International.

Since I was a little child I have been interested in mathematical problems and puzzles. There was always a battle between a friend of mine and me to be number one at the annual mathematics challenge (Kangeroe wedstrijd) of our secundary school. In the first year of secundary school I already knew I wanted to do a study with mathematics in it, and so I did. The further I came in the study the more I knew AM was the right study, but the less certain I became about the master track: Discrete Mathematics and Mathematical Programming (DMMP) or SOR. Courses were chosen that fitted in both tracks. I began with DMMP, switched to SOR and the final project has more to do with DMMP. I think the conclusion must be that mathematics is beautiful and both tracks are interesting.

The website of Omnitrans International cought my attention and luckily they invited me over after they read my letter of application. Although I had no experience in their field, the project is interesting en challenging. Doing my final project here has been a nice and valuable experience.

One cannot do a final project alone. I want to thank Maarten and Werner for their time, expertise and guidance. It was a pleasure to have you as my supervisors. Further I want to thank Michiel for his interest in my work and the other professors, Georg and Richard, for making my final project possible. And let's not forget to thank all colleagues who answered my questions and made Omnitrans Interanational an enjoyable company. Finally of course all my friends and family who supported me. Special thanks to Erik who is always there for me, even when I was moody when something did not work out as planned.

Deventer/Enschede, Maart 2011
Saskia Ton

Notation

Symbol	Meaning
α	empirical determined coefficients
β	empirical determined coefficients
γ	constant used to define a flow-effective PAS
δ_j	equals 1 if $\tilde{C}_i^s = c_{ij} + \tilde{C}_j^s$ and 0 otherwise
$\delta_{ij}^{\{p\}}$	equals 1 if path p contains link ij and 0 otherwise
Δ	path-edge incidence matrix
η	constant used to define a cost-effective PAS
κ	constant
$\mu^{(m)}$	with m={1,2,3} Lagrangian multiplier
ϵ	small number
ϵ_m	with m={2,3,4} small number
λ	scalar
λ^*	optimal scalar
λ_n	scalar at iteration n
Λ	path-OD incidence matrix
σ	vector where $\lvert\sigma_{rs}^T\rvert = [1, 1, \cdots, 1] = m$ when there are m path between OD-pair rs.
Φ_n	thresholds at iteration n
$\xi(f_{ij}^{r;\bullet}, rc_{ij}^{r;\bullet})$	decrease of the objective function
A	set of all links
$B(_r)$	bush of origin r
$B(^s)$	bush of destination s
c_{ij}	cost of link ij
$c_{i\tilde{i}j}$	cost of the turn $i\tilde{i}j$
c_{ij}^*	cost of link ij in UE-assignment
$c_{ij}(f_{ij})$	cost of link ij when it contains flow f_{ij}
$c_{\{p\}}$	cost of a path p
C_i^j	minimum cost of a path from node i to node j
$\tilde{C}_i^s$	cost from node i to destination s
$\bar{c}_{r,s}$	average cost of OD-pair rs
$c^{(n)}$	cost of iteration n
$\mathbf{D}$	vector with demands
$D_{r,s}$	demand from origin r to destination s
e_{ij}	alternative flow on link ij
$e_{ij}^{\bullet,s}$	alternative flow through link ij with destination s
e_i^s	alternative flow leaving node i with destination s
$\mathbf{e}^{(n)}$	alternative flow on iteration n
$e_{\{p\}}$	alterantive flow for path p
f_{ij}	flow on link ij
$\mathbf{f}$	vector of link-flows
$\hat{\mathbf{f}}$	vector of link-flows
$\tilde{\mathbf{f}}$	vector of link-flows

Symbol	Meaning
$\bar{f}_{ij}^{r,s}$	flow on link ij of all OD-pair except rs
f_i	flow leaving node i
f_i^s	flow leaving node i with destination s
$f_{ij}^{r,\bullet}$	flow through link ij with origin r
$f_{ij}^{\bullet,s}$	flow through link ij with destination s
$f_{\{p\}}$	flow on path p
$f_{i\bar{i}j}$	flow on turn $i\bar{i}j$
$\mathbf{f_{\{p\}}}$	vector of path-flows
$f_{ij}^{r,s}$	flow through link ij from origin r to destination s
$\mathbf{f}^{(n)}$	flow at iteration n in terms of link-flow
$f_{ij}^{(n)}$	flow on link ij at iteration n
F	set of feasible flows in terms of link-flow
$F_{\{p\}}$	set of feasible flows in terms of path-flow
FSB(i,s)	containing all nodes j which satisfy $i-j-...-s$
g_{ij}	derivative of the cost at link ij
$g_{\{p\}}$	descent direction of path p
$G_{ij}^{r,s}$	gradient of a link ij caused by OD-pair rs
$\tilde{G}_i^s$	derivative of $\tilde{C}_i^s$
i	node
$\tilde{i}$	node
$\hat{i}$	node
I	set of all nodes
ij	link from node i to node j
j	node
$\tilde{j}$	node
J	set of all nodes in the current forward star bush (FSB)
l	number
l_0	number
n	iteration number
$\tilde{n}$	iteration number
N	maximum number of iterations
$O(\mathbf{f})$	objective function with flow $\mathbf{f}$
p	path
$\hat{p}$	path
P	set of all paths
$P_{r,s}$	set of all directed paths for an OD-pair rs
$P_{r,s}^+$	set of all directed paths for OD-pair rs that carry flow
Q_{ij}	capacity of link ij
r	origin (resource)
$\bar{r}$	origin
R	set of all origins

Symbol	Meaning
rc_{ij}	reduced cost for link ij
$rc_{ij}^{r,\bullet}$	reduced cost origin r, link ij
$rc_{ij}^{r,\bullet}(\mathbf{f})$	reduced cost origin r, link ij flow $\mathbf{f}$
s	destination (sink)
$\bar{s}$	destination
S	set of all destinations
t_{ij}	travel time on link ij
$t_{ij}(f_{ij})$	travel time on link ij when there is f_{ij} flow
$T_{ij}^{scenery}$	"cost" of scenery on link ij
T_{ij}^{fuel}	cost of fuel on link ij
$T_{ij}^{remaining}$	cost of remaining influences on link ij
V_i^s	number used to calculate e_{ij}^s $\forall j \in$FSB(i,s)
w	dummy variable for integration
x	amount of flow
y_{ij}^s	fraction of flow which leave node i going to j

Contents

1 Introduction

In this chapter we begin with some background information. Next we define the problem definition and we end with the structure of this thesis.

1.1 Background

Some responsibilities of local governments are maintenance of the infrastructure and building residential areas. New residential areas cause people to move from/to these areas, changing the flow of traffic. Local governments want to know the influence on the roads caused by these changes to anticipate on the new situation. New traffic situations can be modelled by assigning travellers to a route, assuming the origin and destination of travellers are known. The assignment of travellers on a network is called "traffic assignment problem" (TAP) and can be done in the software "OmniTRANS". As input information is needed the velocities in the network and the number of people departing from and arriving at a zone. The output contains information about the velocity and the number of travellers on a road. This thesis focuses on static assignment, which means that the travellers demand is time independent. We can interpret the static assignment as the flows that would appear if the traffic demand would be constant for a long time. We look for a user equilibrium (UE), meaning that we assume that each traveller minimizes his own cost.

1.2 Problem definition

OmniTRANS uses iterative algorithms with bad convergence; the convergence decreases when the solution gets closer to equilibrium. The algorithms used by OmniTRANS - "incremental assignment", "method of successive averages" and the "Frank-Wolfe" method - were used by almost all TAP-solving programs ten years ago. Substantial improvements in computation times and convergence for finding the UE-assignment were made in the last five years, which caught the attention of Omnitrans International, the developer of OmniTRANS. Since most algorithms are examined exhaustively nowadays, the company is searching for a new algorithm. Newer algorithms store more information and use more memory, but the improvement on computer memory makes these algorithms interesting for Omnitrans International. The algorithms investigated in this thesis are "incremental assignment" (IA) [1], "method of successive averages" (MSA) [1], "Frank-Wolfe" (FW) [2], "simplicial decomposition" (SD) [2], "projected gradient method" (PG) [3], "linear user cost equilibrium" (LUCE) [4], "algorithm B" (Alg. B) [5] and "traffic assignment on paired alternative segments" (TAPAS) [6].
We will compare these algorithms by means of memory requirements, convergence rate and proportionality. With these results we will design a new algorithm based on the requirements of Omnitrans International.

1.3 Structure of this thesis

The rest of this thesis is structured as follows. Chapter 2 explains the TAP in greater detail. A description of the properties can be found in Chapter 3. In Chapter 4 the eight algorithms mentioned above are described, an example is provided and the properties are determined. The modified algorithm can be found in Chapter 5. The result are stated in Chapter 6. In Chapter 7 we can find the conclusion and recommendations.

2 Static user equilibrium

In this chapter the TAP is explained in greater detail and in a mathematical formulation, an often used cost function is described, next it is explained how we can add junction modelling and the section ends with a proof of the equivalence of two mathematical descriptions, the existence and uniqueness of a UE on a network without junction modelling and under certain conditions on a network with junction modelling.

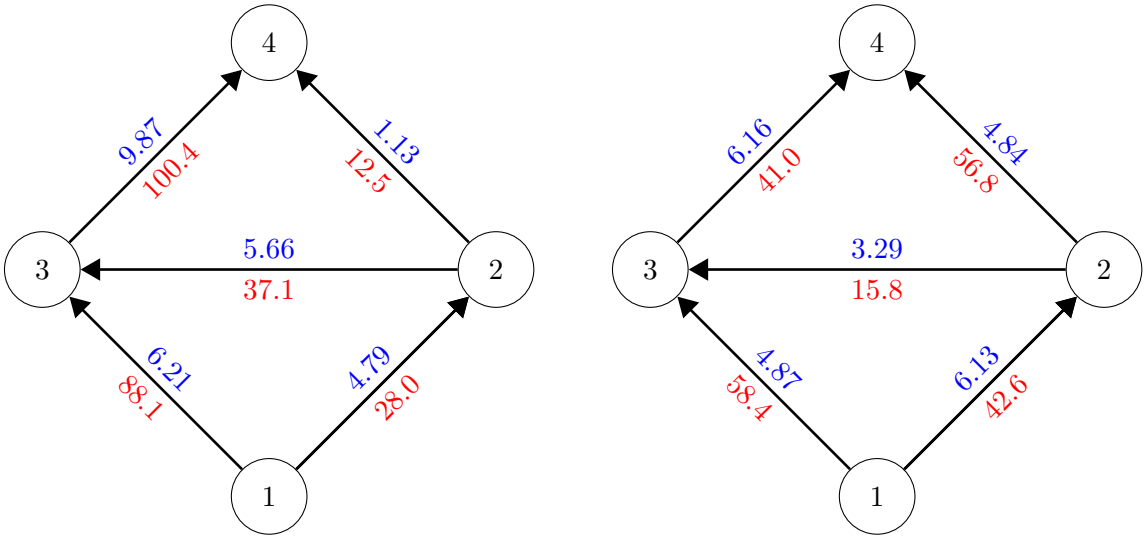
2.1 Mathematical formulation

We begin with the notation and explaining some terminology. A network consists of nodes (areas), links (streets or lanes) and flow (travellers). A link between node i and j is called ij. The flow on link ij is f_{ij}, the cost c_{ij} and an alternative flow e_{ij}. The cost on a link is only dependent on the flow on that link and the cost of a path is the sum of the costs of all links in that path. The flow on path p is $f_{\{p\}}$, the cost of this path is $c_{\{p\}}$ and an alternative flow on this path is $e_{\{p\}}$. Origin $r \in R$ is the node where a flow begins and destination $s \in S$ the node where a flow ends, where R is the set of all origins and S is the set of all destinations. The set of all paths is P and the set of all paths with origin r and destination s is $P_{r,s}$. A path $p \in P_{r,s}$ is thus a path with $r - i - ... - s$. The objective function $O(\mathbf{f})$ is depending on the flow $\mathbf{f}$. $F_{\{p\}}$ is the set of feasible path-flows and F the set of feasible link-flows. $\delta_{ij}^{\{p\}}$ is 1 if link ij is part of path p and 0 otherwise. The cost of a minimum cost path is C_r^s.
Our problem is finding the flow which satisfies all demands in such a way that each traveller minimizes his own cost, where we assume the demand $D_{r,s}$ is known in advance.

To show what we mean by a UE we will show an unequilibrated network and a equilibrated network. We will first confirm that Figure 2.1.1a is not equilibrated. The demand from node 2 to node 4 is 2, where the flow on path $2-4$ is 1.13. This means that node 2 uses path $2-3-4$ while path $2-4$ is cheaper. The travellers of OD-pair 24 can lower their cost, which means that the network is not in equilibrium. Figure 2.1.1b is in equilibrium, since all paths between every OD-pair are equal. We have $c_{\{1-2-3\}} = c_{\{1-3\}}$, $c_{\{1-2-3-4\}} = c_{\{1-2-4\}} = c_{\{1-3-4\}}$ and $c_{\{2-3-4\}} = c_{\{2-4\}}$.

All travellers from a certain origin to a certain destination are called travellers of an origin-destination-pair, or OD-pair. We assume that the demand of every OD-pair is known in advance and that travellers cannot change their path during their journey. In a UE all travellers minimize their own cost, leading to equal cost of each path used by that OD-pair. It is proved that the cost of this path is equal to the minimum cost path of that OD-pair. We can prove this by contradiction. Suppose there is a path used with a higher cost then the cheapest path. The traveller on this path can lower his cost by changing to the cheapest path. This will be repeated until there is no used path costlier then the cheapest one. But then there is no path more expensive then the cheapest one, meaning all used path have equal cost, namely equal to the cheapest path possible from the origin to the destination.

In case of link-flows we define $f_{ij} = \sum_{p \in P} \delta_{ij}^{\{p\}} f_{\{p\}} \; \forall ij \in A$.
We want that the cost of every used path is equal to the minimum cost path of an OD-pair, the flow on a path is non-negative and the demand-requirement is met.



(a) An unequilibrated network. (b) An equilibrated network.

Figure 2.1.1: The network of Figure 4.0.1, without the junction, with flow that does not equilibrate the cost in (a) and flow that does equilibrate the cost of (b). The blue number above a link is the flow and the red number below a link is the cost of that link.

This is written down in 1952 in Wardrop principles [7]:

$$\begin{array}{ll} f_{\{p\}}\,(c_{\{p\}} - C_r^s) = 0 & \forall\, p \in P_{r,s} \\ c_{\{p\}} \geq C_r^s, & \forall\, p \in P_{r,s} \\ f_{\{p\}} \geq 0, & \forall\, p \in P \\ \sum_{p \in P_{r,s}} f_{\{p\}} = D_{r,s} & \forall\, r \in R,\, s \in S \end{array}$$

The solution of the following variational inequality (VI) problem describes a Wardrop UE. We want to find flows $f_{\{p\}} \in F_{\{p\}}$ such that

$$\sum_{p \in P} c_{\{p\}}(f_{\{p\}}) \cdot (e_{\{p\}} - f_{\{p\}}) \geq 0, \quad \forall \mathbf{e} \in F_{\{p\}}$$

When we rewrite this path-based VI problem into a link-based VI problem, we want to find link flows $f_{ij} \in F$ such that

$$\sum_{ij} c_{ij}(f_{ij}) \cdot (e_{ij} - f_{ij}) \geq 0, \quad \forall \mathbf{e}, \mathbf{f} \in F$$

Beckmann, McGuire and Winsten developed in 1956 the formulation of the standard UE problem as a mathematical program under the assumptions of separable additive costs, the cost and flow are at least zero ($c_{ij} \geq 0$, $f_{ij} \geq 0$) and the cost of a link is dependent of the flow on this link ($c_{ij}(f_{ij})$). Existence and uniqueness are proven for Beckmann's formulation [8].

The following is Beckmann's formulation:

$$\begin{array}{rll} \min\limits_{f} & \mathrm{O}(\mathbf{f}) \;=\; \sum_{ij\in A} \int_{w=0}^{f_{ij}} c_{ij}(w)\,\mathrm{d}w & \\ \text{subject to} & f_{ij} = \sum_{p\in P} f_{\{p\}}\delta_{ij}^{\{p\}} & \forall ij \in A \\ & \sum_{p\in P_{rs}} f_{\{p\}} = D_{r,s} & \forall r \in R,\ s \in S \\ & f_{\{p\}} \geq 0 & \forall p \in P \end{array}$$

When we would have the objective function $\min\limits_{f} \sum\limits_{ij\in A} f_{ij} \cdot c_{ij}$ we would find the system optimum instead of the user equilibrium. The system optimum is a flow where the total cost of all travellers together is minimal.

It can happen that two quite similar algorithms are developed at the same time without knowlegde of each others algorithm, but many formulations and algorithms are based on earlier papers. Although there are many more algorithms and maybe other formulations, we have made a timeline for the formulations and algorithms used in this paper, which can be found in Figure 2.1.2.

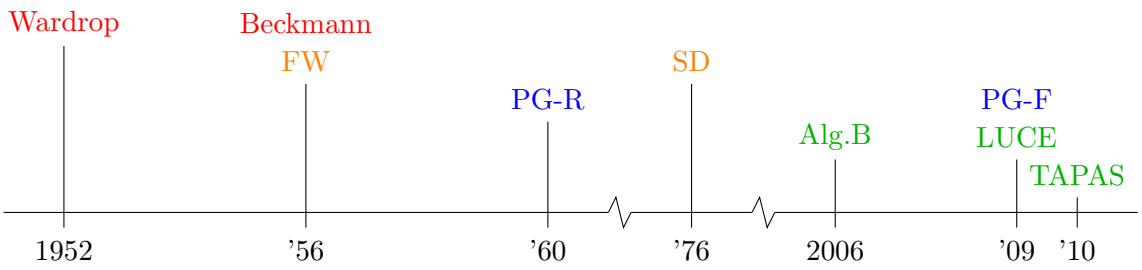


Figure 2.1.2: Timeline of the publications of the algorithms and formulations. PG-R is the original projected gradient method of Rosen [9] and PG-F is the PG method of Florian et al [3]. All other abbreviations are the algorithms described in this thesis. Red are mathematical formulations, orange abbreviations are link-based, blue are path-based and green are bush-based.

2.2 BPR-function

Now that the formulations are known, we continue with an explanation of the links and the cost of these links.
A road consists of at least one traffic lane. Different traffic lanes can have different properties, like maximum velocity and permitted vehicles. But a link in a network can only have one specification. So it is, for example, a highway, accessible for all traffic and a maximum velocity of 100 km/h or a bicycle path, only accessible for bicycles and a maximum velocity of 25 km/h. If different lanes have different specifications they are modelled as two separate links.
The relation between load and travel time is represented by a time loss function. The Bureau of Public Roads function, or shortly BPR-function, is used most often.

The BPR-function has the following formulation: [10]

$$t_{ij} = t_{ij}(0)\,(1 + \alpha \left(\frac{f_{ij}}{Q_{ij}}\right)^{\beta})$$

t_{ij} travel time on link ij
$t_{ij}(0)$ travel time on link ij in an unloaded network
Q_{ij} maximum capacity of link ij
α, β empirical determined coefficients
f_{ij} flow on link ij

The often chosen values for α and β give a function which is slightly increasing at the beginning but with a dramatic increase when the flow approaches full capacity. Two BPR-functions for $\beta = 4.0$ are plotted in Figure 2.2.1. The x-axis goes beyond 1 since some algorithms send more than a 100% of the capacity through a link during the iterations.

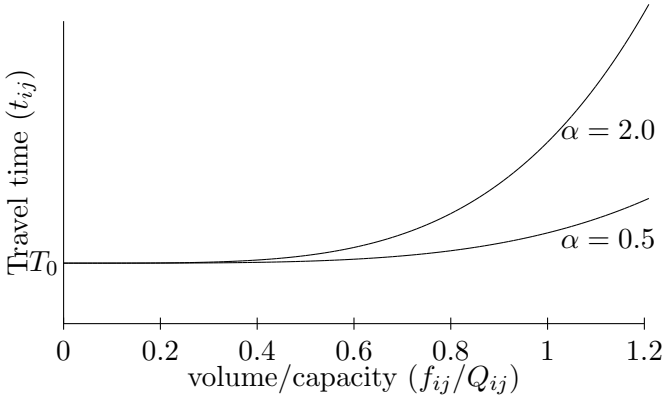


Figure 2.2.1: The BPR-function for $\beta = 4$

With the BPR-function we can calculate the travel time. Previously however cost was mentioned instead of time. Which paths travellers choose is not only dependent on the travel time, but also of, for example, scenery, travel distance or number of traffic-lights. All these factors are considered and put together to yield the generalized cost. The travel time, and thus the BPR-function, plays an important role in calculating the generalized cost. An example of the generalized cost as $c_{ij} = t_{ij} + T_{ij}^{scenery} + T_{ij}^{fuel} + T_{ij}^{remaining}$, where $T_{ij}^{scenery}$ is the "cost" of scenery, T_{ij}^{fuel} the cost of fuel and $T_{ij}^{remaining}$ the cost of the remaining influences on link ij. Finding the UE is hard, causing the methods to be iterative instead of solving the problem at once. All methods, except incremental assignment, always converge to a UE, but since this can take very long or exceeds computer precision we end up with an approximated UE-assignment.

In our example networks we will use BPR-functions. When we look at the cost for link 12 in Figure 4.0.1 we have $c_{12} = t_{12}(0)\,(1+\alpha \left(\frac{f_{12}}{Q_{12}}\right)^{\beta}) = 5\,(1+\frac{1}{5}\left(\frac{f_{12}}{1}\right)^{2}) = 5+f_{12}^2$. We can interpret the cost function in the following way: going from node 1 to node 2 costs $5 + f_{12}^2 = 5 + 0^2 = 5$ if there is zero flow. When there is a flow of 3, the cost of using this street becomes $5+3^2 = 14$

due to congestion.

2.3 Junction modelling

When thinking about traffic lights, we think about the green light we just missed and the extra travel time this adds to our journey. Traffic-lights or, more general, a junction may lead to extra costs. The extra cost caused by crossing roads is strongly preferred to be taken into account when calculating the costs.
A junction may also change the maximum capacity of the links connected to it. Suppose we have a link with a maximum capacity of 1000 vehicles per hour. When there is a traffic light on this road with a green-time of 40% for turning right, then we have a maximum capacity of $0.40 \cdot 1000 = 400$ vehicles per hour for that link turning right.

When we do not include junction modelling we have a network of directed arcs where the cost of a route is the sum of the costs of the links it uses, where the cost of a link is dependent on the flow on that link. In junction modelling we can not fulfil this assumption, leading to a network that may have no unique UE or even a network that has no UE at all. Most real-life networks with junction modelling however do have a UE.

When we add junctions to a network we "expand" the junction. This means that we add nodes at every ramp and exit and add a link for every turn. We can see an expanded junction in Figure 2.3.1 where one can go left, right and straight on. To mark the difference between old links and new links, we will call the new links "turns" and the costs of turns are "turns costs".

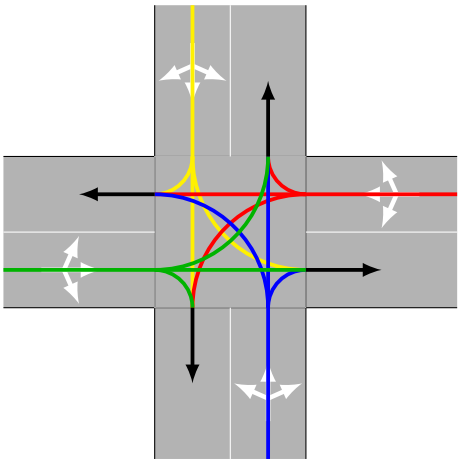


Figure 2.3.1: Expanded junction. The colors are added to make clear which turns one can make when one arrives from a certain link.

By expanding the junction, we get a new network where some links are only depent on the flow on that link (the old links) and some are also dependent on the flow on other links (the turns). With a little adaptation, all algorithms described in this paper can be used on a network with junction modelling, although we have to be careful when saying something

about the convergence rate. The adaptation can be that during an iteration the cost of a turn is assumed to be constant, which is described into more detail in Chapter 5.3.

2.4 Theoretical background

For completeness we will prove some theorems in the next section. We will first prove that Beckmann's formulation and Wardrop's UE are equivalent. Next we prove the existence of a UE in a network without junctions and provide a condition for the existence of a UE in a network with junction modelling. Then we prove that in networks without junction modelling there is, under strict monotonicity on c, a unique solution in terms of link flow and provide a condition for a unique UE for the case when there is junction modelling.

2.4.1 Equivalence Beckmann and UE

We will prove in this subsection that searching for an $\mathbf{f}$ that minimizes Beckmann's minimization is equivalent with searching for an $\mathbf{f}$ that satisfies Wardrop's conditions under monotonicity of $c_{ij}(f_{ij})$.

For convenience we will introduce some matrices: Λ is the path-OD incidence matrix and Δ is the path-edge incidence matrix. When we recall Beckmann's formulation with the old and the new notation we get

$$\begin{aligned} \min_{\mathbf{f}} \quad & \mathrm{O}(\mathbf{f}) = \textstyle\sum_{ij\in A} \int_{w=0}^{f_{ij}} c_{ij}(w)\,\mathrm{d}w & & \\ \text{subject to} \quad & \Delta\mathbf{f_{\{p\}}} = \mathbf{f} & & \textstyle\sum_p \delta_{ij}^{\{p\}} f_{\{p\}} = f_{ij} \\ & \Lambda\mathbf{f_{\{p\}}} = \mathbf{D} & & \textstyle\sum_p \delta_{\{p\}}^{r,s} f_{\{p\}} = D_{r,s} \\ & \mathbf{f_{\{p\}}} \geq \mathbf{0} & & f_{\{p\}} \geq 0 \end{aligned} \tag{1}$$

The Lagrangian of this minimization problem can be formulated as

$$\begin{aligned} L(\mathbf{f},\mathbf{f_{\{p\}}},\mu^{(2)},\mu^{(3)},\mu^{(1)}) &= O(\mathbf{f}) - (\mu^{(2)})^T(\Lambda\mathbf{f_{\{p\}}} - D) + (\mu^{(3)})^T(\Delta\mathbf{f_{\{p\}}} - \mathbf{f}) - (\mu^{(1)})^T\mathbf{f_{\{p\}}} \\ \text{subject to } \mathbf{f_{\{p\}}} &\geq 0 \\ \mu^{(1)} &\geq 0, \end{aligned}$$

where $\mu^{(1)}$, $\mu^{(2)}$ and $\mu^{(3)}$ are Lagrange multipliers.

The Karush-Kuhn-Tucker (KKT) conditions for this minimization are

$\partial L/\partial \mathbf{f} = 0,\ \partial L/\partial \mathbf{f_{\{p\}}} = 0,\ \mu^{(1)}\mathbf{f_{\{p\}}} = 0,\ \mathbf{f_{\{p\}}} \geq 0$ and $\mu^{(1)} \geq 0$

When we write out the first two conditions we get

$\partial L/\partial \mathbf{f} = \partial O(\mathbf{f})/\partial \mathbf{f} - \mu^{(3)} = 0$
$\partial L/\partial \mathbf{f_{\{p\}}} = -\Lambda^T\mu^{(2)} + \Delta^T\mu^{(3)} - \mu^{(1)} = 0$

We can fill in $\partial O(\mathbf{f})/\partial \mathbf{f} = c(\mathbf{f})$, which gives $c(\mathbf{f}) = \mu^{(3)}$
$\Lambda^T\mu^{(2)} - \Delta^T c(\mathbf{f}) + \mu^{(1)} = 0$
$\mu^{(1)}\,\mathbf{f_{\{p\}}} = 0$ (2)
$\mathbf{f_{\{p\}}} \geq 0$

$\mu^{(1)} \geq 0$

Note that $\Lambda_{p,rs} = \begin{cases} 1 & \text{if path p} \in P_{rs} \\ 0 & \text{otherwise} \end{cases}$

And thus

$$\Lambda^T \mu^{(2)} = \begin{bmatrix} \sigma_1 & 0 & 0 & 0 & \cdots \\ 0 & \sigma_2 & 0 & 0 & \cdots \\ 0 & 0 & \sigma_3 & 0 & \cdots \\ \vdots & \vdots & \ddots & \ddots & \\ 0 & 0 & 0 & 0 & \sigma_{|OD|} \end{bmatrix} \begin{bmatrix} \mu_1^{(2)} \\ \mu_2^{(2)} \\ \mu_3^{(2)} \\ \vdots \\ \mu_{|P|}^{(2)} \end{bmatrix} = \begin{bmatrix} \mu_1^{(2)}\sigma_1 \\ \mu_2^{(2)}\sigma_2 \\ \mu_3^{(2)}\sigma_3 \\ \vdots \\ \mu_{|P|}^{(2)}\sigma_{|P|} \end{bmatrix}$$

where $\sigma_{rs}^T = \begin{bmatrix} 1 & \cdots & 1 \end{bmatrix}$ with $|\sigma_{rs}| = m$ if OD-pair rs has m paths.

When we rewrite the formula's we get $\mu^{(2)} = \Delta^T c(\mathbf{f}) - \mu^{(1)}$. When we write this out we get $\mu_{rs}^{(2)} = [\Delta^T c(f_{ij})]_p - \mu_p^{(1)} = \begin{cases} = c_{\{p\}}(\mathbf{f}) & \text{if } f_{\{p\}} > 0 \\ \leq c_{\{p\}}(\mathbf{f}) & \text{if } f_{\{p\}} = 0 \end{cases}$

This means that a path between OD-pair rs has the same costs as the minimum cost path of rs if it contains flow and has equal or higher cost when the path does not contain flow. This is exactly the description of a Wardrop UE. We can see that when we fill in $\mu_{rs}^{(2)} = C_r^s$ we can reformulate the above to $f_{\{p\}}\,(c_{\{p\}} - C_r^s) = 0$ and $c_{\{p\}} \geq C_r^s$, $\forall\ p \in P_{r,s}$, which are Wardop's conditions. One can remark that Wardrop has also the conditions $f_{\{p\}} \geq 0$ and $\sum_{p \in P_{r,s}} f_{\{p\}} = D_{r,s}$, but these are also conditions in Beckmann's formulation.

From the above argumentation and using standard results in optimization we obtain

Theorem 1
For a feasible edge flow vector $\mathbf{f}$ with corresponding path-flow vector $\mathbf{f_{\{p\}}}$ the following holds
(a) $\mathbf{f}$ is a Wardrop UE if and only if $\mathbf{f}$ (and thus $\mathbf{f_{\{p\}}}$) satisfies the KKT conditions (2).
(b) If $\mathbf{f}$ is a minimizer of (1) then it satisfies the KKT conditions (2), and by **(a)** it is a Wardrop UE.
(c) If O(F) is convex and $\mathbf{f}$ ($\mathbf{f_{\{p\}}}$) satisfy the KKT conditions (2) then $\mathbf{f}$ ($\mathbf{f_{\{p\}}}$) is a minimizer of (1).

2.4.2 Existence UE

In this subsection we will give conditions for the existence of a UE, which are satisfied by networks without junction modelling.
Since the feasible set of (1) is compact and bounded we can obtain from the Weierstrass (existence) theorem:

Theorem 2
If the function $O(\mathbf{f})$ in (1) is continuous on the feasible set of (1), then there exists at least one minimizer $\tilde{\mathbf{f}}$ of (1).

Note that by Theorem 1(b) $\tilde{\mathbf{f}}$ is a Wardrop UE.

We can remark from Theorem 2 that the same holds for any cost function $c(\mathbf{f})$ such that $c(\mathbf{f}) = \nabla\tilde{O}(\mathbf{f})$ for some function $\tilde{O}$. By Poincaré's lemma [11] such a function $\tilde{O}$ exists if $c(\mathbf{f})$ is C^1 and the conditions $\frac{\partial c_{ij}(\mathbf{f})}{\partial f_{\tilde{i},\tilde{j}}} = \frac{\partial c_{\tilde{i},\tilde{j}}(\mathbf{f})}{\partial f_{i,j}}$ hold.

For in the general case we can use the lemma of Hartman and Stampacchia's lemma [12].

Hartman and Stampacchia's lemma
If $c(\mathbf{f})$ is continuous then there exist a feasible flow $\tilde{\mathbf{f}}$ such that $c(\tilde{\mathbf{f}})(\mathbf{f} - \tilde{\mathbf{f}}) \geq 0 \; \forall \mathbf{f} \in F$.

When we combine Stampacchia's lemma with variational inequality, we get that if $c(\mathbf{f})$ is continuous $\tilde{\mathbf{f}}$ is a UE.

2.4.3 Uniqueness UE

In this subsection we will give conditions for the uniqueness of a UE, which are satisfied by networks without junction modelling.
The uniqueness of a Wardrop UE depends on a monotonicity condition for the cost function $c(\mathbf{f})$.

Theorem 3
If $c(\mathbf{f})$ satisfy $[c(\tilde{\mathbf{f}}) - c(\mathbf{f})]^T(\tilde{\mathbf{f}} - \mathbf{f}) > 0 \;\; \forall \mathbf{f} \neq \tilde{\mathbf{f}}, \;\; \tilde{\mathbf{f}}, \mathbf{f} \in F$ then $\tilde{\mathbf{f}}$ is a unique Wardrop equilibrium.

We want to show that if we assume that $\hat{\mathbf{f}}$ and $\tilde{\mathbf{f}}$ are distinct equilibrium flows this leads to a contradiction, and thus the equilibrium flow must be unique.

Suppose there are two equilibrium flows $\hat{\mathbf{f}}$ and $\tilde{\mathbf{f}}$ where $\hat{\mathbf{f}} \neq \tilde{\mathbf{f}}$.
We know from variational inequality that $c(\tilde{\mathbf{f}})^T(\mathbf{f} - \tilde{\mathbf{f}}) \geq 0 \;\; \forall \mathbf{f} \in F$, so this is also true for $\mathbf{f} = \hat{\mathbf{f}}$, giving $c(\tilde{\mathbf{f}})^T(\hat{\mathbf{f}} - \tilde{\mathbf{f}}) \geq 0$. The same reasoning can be used for the other equilibrium flow, resulting in $c(\hat{\mathbf{f}})^T(\tilde{\mathbf{f}} - \hat{\mathbf{f}}) \geq 0$.

When we sum up these two expressions we get $c(\tilde{\mathbf{f}})^T(\hat{\mathbf{f}} - \tilde{\mathbf{f}}) + c(\hat{\mathbf{f}})^T(\tilde{\mathbf{f}} - \hat{\mathbf{f}}) \geq 0$, which can be rewritten as $[c(\tilde{\mathbf{f}}) - c(\hat{\mathbf{f}})]^T(\hat{\mathbf{f}} - \tilde{\mathbf{f}}) \geq 0$.
This is the same as $[c(\tilde{\mathbf{f}}) - c(\hat{\mathbf{f}})]^T(\tilde{\mathbf{f}} - \hat{\mathbf{f}}) \leq 0$. This is a contradiction with $[c(\tilde{\mathbf{f}}) - c(\hat{\mathbf{f}})]^T(\tilde{\mathbf{f}} - \hat{\mathbf{f}}) > 0$.
We can conclude from this that our assumption, the existence of 2 equilibrium flows, is not true if $[c(\tilde{\mathbf{f}}) - c(\mathbf{f})]^T(\tilde{\mathbf{f}} - \mathbf{f}) > 0$. We already proved that there exists an equilibrium flow, thus the equilibrium flow must be unique.

3 Properties

At the end of every paragraph in Section 4 there is a table with an overview of four properties; the property table. First we will explain how to interpret this table, next we explain the properties.

As example of a property table we will show here is the table of MSA:

Memory usage	++
Proportionality	+
Rate of convergence	−

We have chosen for the three properties "memory usage", "proportionality" and "rate of convergence" since Omnitrans International is interested in these and we believe we need these properties to distinguish the algorithms properly.

A + is a desirable and a − is less desirable, but the exact explanation of the used signs are explained in the following three tables.

Memory usage

Sign	Used when the algorithm is
++	link-based
+	link-based and stores multiple iterations
+−	bush-based
−	bush-based and stores PASs between iterations
−−	path-based

Proportionality

Sign	Used when the algorithm
+	satisfies proportionality condition
−	does not satify proportionality condition

Rate of convergence

Sign	Used when rate of convergence is
+	high
+−	average
−	low

3.1 Memory usage

The memory usage of the algorithms differs greatly, mostly caused by the aggregation level. A second cause of larger memory usage is the storage of "pairs of alternative segments" (PASs). First the different levels of aggregation are explained, next the PASs.

Level of aggregation

There are three levels of aggregation: link-, path- and bush-based.
A link-based algorithm stores the total amount of flow through a link. An example of a link-based assignment can be found in Figure 3.1.1.

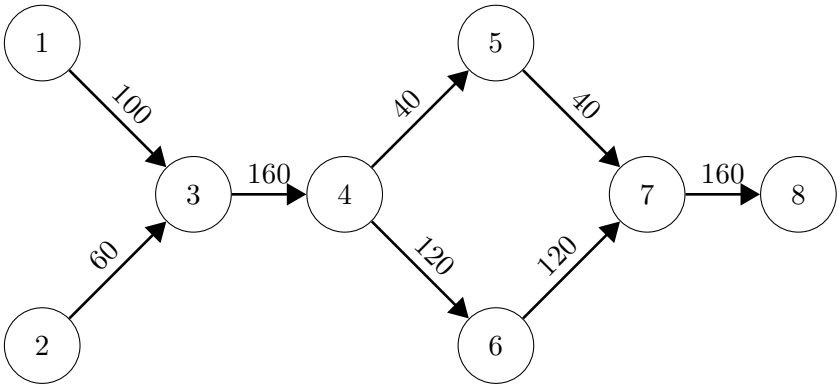


Figure 3.1.1: Link-based assignment on a network with demands $D_{1,8} = 100$ and $D_{2,8} = 60$ travellers. The origins are 1 and 2 and the destination is 8. The circles are nodes, the arcs are links and the number above a link is the flow on that link.

It is also possible to store every used path. Figure 3.1.2 is an example of a path-based assignment on the same network as Figure 3.1.1.

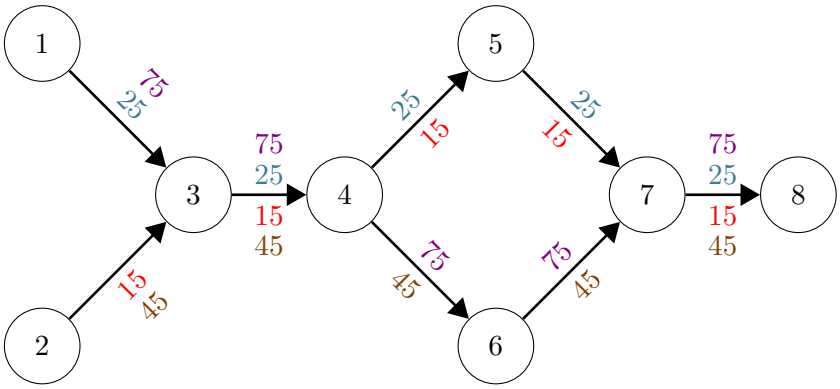


Figure 3.1.2: Path-based assignment with proportionality on a network with demands $D_{1,8} = 100$ and $D_{2,8} = 60$ travellers. The purple number above a link is the amount of flow of path $1-3-4-6-7-8$ on this link, the gray number of path $1-3-4-5-7-8$, the red number of path $2-3-4-5-7-8$ and the brown number is of path $2-3-4-6-7-8$.

A bush is a set of predetermined paths. Two examples are an origin-based bush, where a set consists of all paths with origin r, and a destination-based bush, where a set consists of all paths with destination s. An origin-based assignment can be found in Figure 3.1.3. Other special cases of bush-based are path-based, where every path is a set, and link-based, where the entire network is one set.

Pair of alternative segments (PAS)

The algorithms Alg. B and TAPAS use PASs to converge to a UE-assignment. A PAS consists of two distinct paths with the same begin node and the same end node and with no

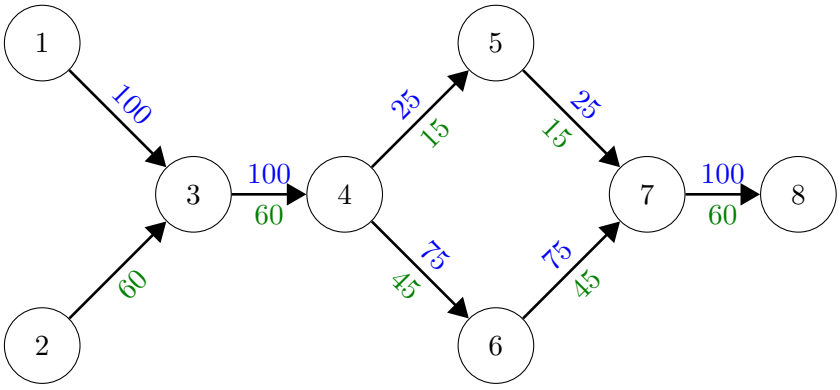


Figure 3.1.3: Origin-based (bush-based) assignment with proportionality on a network with demands $D_{1,8} = 100$ and $D_{2,8} = 60$ travellers. The blue number above a link is the amount of flow with origin 1 on this link and the green number below a link is the amount of flow with origin 2 on this link.

common nodes in between. Looking at Figure 3.1.4 we can see one PAS. One segment of this PAS is outlined by a blue arrow and the other segment is outlined by a red arrow. These two segments are together one PAS.

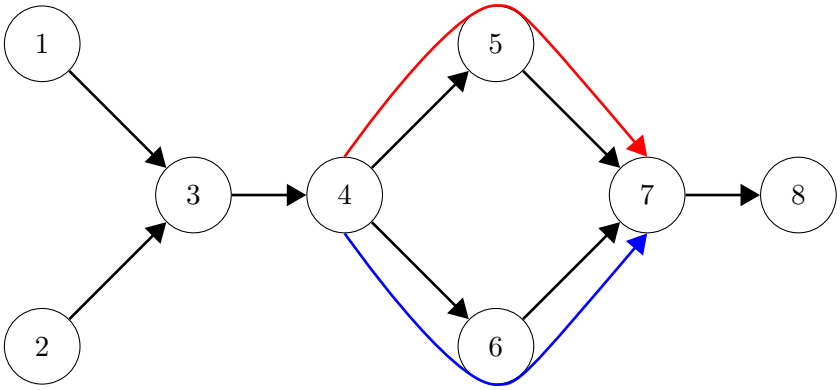
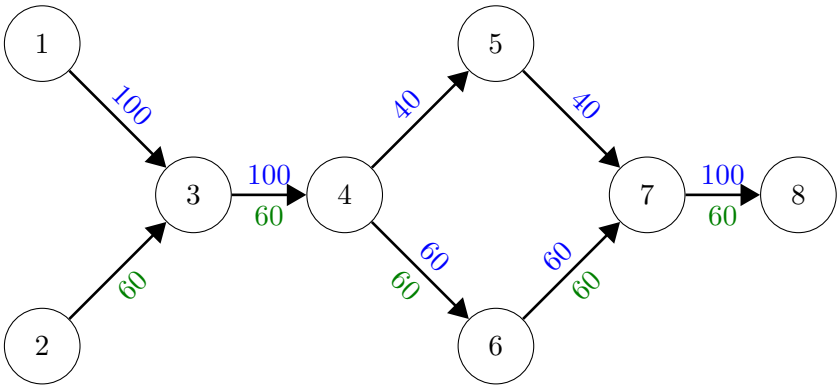


Figure 3.1.4: An example of a PAS. The blue segment (4−6−7) and the red segment (4−5−7) are together one PAS.

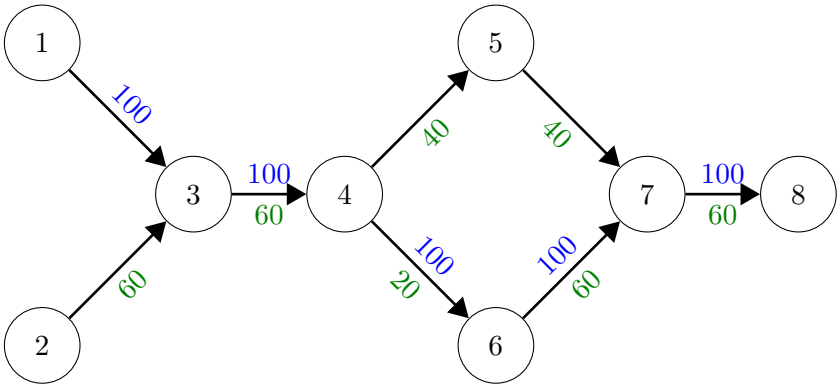
Alg. B determines a few PASs every iteration and forgets these PASs at the end of the iteration. The storage of PASs is not increasing the memory usage much. TAPAS eliminates a PAS if it is not used in the last 3 iterations, potentially increasing the memory usage much. It depends on the network what percentage of memory usage is needed to store PASs. Some numbers from the paper [6] are that for Chicago regional there is an memory usage of 607.51 MB of which is 532.85 MB on origin-based link flow and 12.29 MB on storing PASs, and for Berlin center the memory usage is 204.30 MB of which is 8.71 MB on origin-based link flows and 0.15 MB on storing PASs.

3.2 Proportionality

The UE has a unique solution when we look at the amount of flow on a link, but is not unique when we look at the paths travellers take. We can make the assignment unique in terms of path-flow by requiring maximum entropy. An assignment satisfies maximum entropy when the probability distribution of choosing a certain path is the one with the largest entropy. This condition makes sure that travellers of an OD-pair choose all likely paths instead of, for example, using as least different paths as possible. In the ideal case an assignment satisfies maximum entropy, but this is difficult to check. An approximation of entropy maximization is proportionality, which is easy to check. Consider Figure 3.1.1 as an example. There are multiple path flows possible corresponding to these link flows. Two examples are Figure 3.2.1a and Figure 3.2.1b. We neither expect that only origin 1 uses path $4-5-7$ (a) nor only origin 2 (b).



(a) *Path* $4-5-7$ *is only used by origin*1.



(b) *Path* $4-5-7$ *is only used by origin* 2.

Figure 3.2.1: Origin-based (bush-based) assignment without proportionality on a network with demands $D_{1,8} = 100$ *and* $D_{2,8} = 60$ *travellers. The blue number above a link is the amount of flow with origin 1 on this link and the green number below a link is the amount of flow with origin 2 on this link.*

There are 160 vehicles per hour arriving at node 4. A quarter of them uses the upper path and the rest is using the lower path. We expect that a quarter of the 100 vehicles of origin 1 uses the upper part and a quarter of the 60 vehicles of origin 2. Thus for path $4-5-7$ we expect $40/160 \cdot 100 = 25$ vehicles of origin 1 and $40/160 \cdot 60 = 15$ vehicles of origin 2. For

path $4-6-7$ we have $f^{1,8}_{\{4-6-7\}} = 120/60 \cdot 100 = 75$ and $f^{2,8}_{\{4-6-7\}} = 120/60 \cdot 60 = 45$. This is exactly as depicted in Figure 3.1.3, which satisfies proportionality.
We can equilibrate every PAS such that the proportion of flow in one segment is equal to the proportion of flow in the other segment.
In mathematics we have

$$\frac{f^{r,s}_{\{i-..-\tilde{\imath}-..-j\}}}{f^{\tilde{r},\tilde{s}}_{\{i-..-\tilde{\imath}-..-j\}}} = \frac{f^{r,s}_{\{i-..-\hat{\imath}-..-j\}}}{f^{\tilde{r},\tilde{s}}_{\{i-..-\hat{\imath}-..-j\}}}$$

where $i, \tilde{\imath}, \hat{\imath}$ and j are nodes, r and $\tilde{r}$ are origins, s and $\tilde{s}$ are destination and $i-..-\hat{\imath}-..-j$ and $i-..-\tilde{\imath}-..-j$ are both paths in a PAS.

3.3 Rate of convergence

We need a criterion to measure the rate of convergence. One can, for example, do 100 iterations and compare the time it took to do 100 iterations. This is however not a very good manner, since an algorithm may need less iterations then another but the time one iteration takes is higher. A better way is to determine a gap after a certain amount of time or determine the time needed to reach a certain gap. This gap can be defined in several ways. We recall the following notation for defining the gap:
The flow on link ij is f_{ij} and $c_{ij}(f_{ij})$ is the cost of link ij when there is f_{ij} flow. C^j_i is the cost of the minimum cost path between node i and node j and $c_{\{p\}}$ is the cost of path p. $D_{r,s}$ is the demand from origin r to destination s.

A gap can be defined by the following criteria:

- Relative gap (RGAP) $= 1 - \frac{\text{total minimum cost}}{\text{total cost}} = 1 - \frac{\sum_{r\in R}\sum_{s\in S} C^s_r D_{r,s}}{\sum_{ij\in A} f_{ij}\, c_{ij}}$

  This manner is used most often. The advantage of RGAP is that it gives a good measurement of the percentage of reducible costs. This criterion can be used for path-based, bush-based and link-based assignments. It can happen that a small part of the network is very bad, but that the RGAP is small, which is a disadvantage.

- Average excess cost (AEC) $= \frac{\text{total cost}-\text{total minimum cost}}{\text{total demand}}$
  $$= \frac{(\sum_{ij\in A} f_{ij}\, c_{ij}) - (\sum_{r\in R}\sum_{s\in S} C^s_r D_{r,s})}{\sum_{r\in R}\sum_{s\in S} D_{r,s}}$$

  The AEC is a measure of the average reducible cost per demand. This criterion can be used for path-based, bush-based and link-based assignments. It is the RGAP but instead of dividing it by the total cost it is divided by de total demand. This criterion has the same disadvantage as RGAP, since this criterion has also the possibility that a small part of the network is very bad while the AEC is small.

- max-min cost difference (MMCD) $= \max_{p\in P_{r,s}} \{c_{\{p\}} - C^s_r\}$

  Wardrop's principles stated that the maximum used cost and the minimum cost of an

OD-pair must be the same. MMCD is thus directly related to Wardrop's principles. The downside of this criterion is that it needs the minimum cost route and the maximum used cost route, which is not available for link-based assignments.

In this thesis RGAP is used to determine the rate of convergence. The first reason for this is that we want a criterion usable for link-based, bush-based and path-based. The second reason is that AEC and RGAP are closely related, but RGAP is used in most papers.

We have

$$\begin{aligned}
\sum_{ij} f_{ij} c_{ij} &= \sum_{ij} (\sum_{p} \delta_{ij}^{\{p\}} f_{\{p\}}) c_{ij} \\
&= \sum_{ij} \sum_{p} \delta_{ij}^{\{p\}} f_{\{p\}} c_{ij} \\
&= \sum_{p} f_{\{p\}} \sum_{ij} c_{ij} \delta_{ij}^{\{p\}} \\
&= f_{\{p\}} c_{\{p\}}
\end{aligned}$$

and therefore we can write RGAP as $1 - \frac{\sum_{r \in R} \sum_{s \in S} C_r^s D_{r,s}}{\sum_{p \in P} f_{\{p\}}\, c_{\{p\}}}$.
Since $c_{\{p\}} \geq C_r^s$ we have

$$\begin{aligned}
1 - \frac{\sum_{r \in R} \sum_{s \in S} C_r^s D_{r,s}}{\sum_{p \in P} f_{\{p\}}\, c_{\{p\}}} &= 1 - \frac{\sum_{r \in R} \sum_{s \in S} C_r^s D_{r,s}}{\sum_{r \in R} \sum_{s \in S} \sum_{p \in P_{r,s}} f_{\{p\}}\, c_{\{p\}}} \\
&\geq 1 - \frac{\sum_{r \in R} \sum_{s \in S} C_r^s D_{r,s}}{\sum_{r \in R} \sum_{s \in S} \sum_{p \in P_{r,s}} f_{\{p\}}\, C_r^s} \\
&= 1 - \frac{\sum_{r \in R} \sum_{s \in S} C_r^s D_{r,s}}{\sum_{r \in R} \sum_{s \in S} D_{r,s}\, C_r^s} \\
&= 1 - 1 = 0
\end{aligned}$$

When we are close to an equilibrium, we have that $c_{\{p\}}$ and C_r^s are almost equal. The fraction will then be almost 1 and the RGAP close to 0.

4 Algorithms

We begin this chapter with a description of a small example network. Then the "all-or-nothing-assignment" (AON-assignment) is described and an example of this assignment is given. Further are the bush of an origin and a destination given based on the AON-assignment. After that all subsections have the following structure: a small introduction, a description, an example on the network of Figure 4.0.1, where some algorithms use this figure without the junction modelling of node 3, and a summary of the properties.

We will explain first why we have chosen this example network. We wanted a network that is small enough to perform 3 iterations by hand. The basis of this network can be found in the paper of LUCE [4], where the network has 2 origins and 1 destination. To make the differences between the algorithms in this thesis more clear we added an OD-pair to create 1 more destination. The junction is added to show how junction modelling works in the algorithms. The downside of a small network is that some algorithms may look like each other since the assignments of the first one or two iterations can be the same.

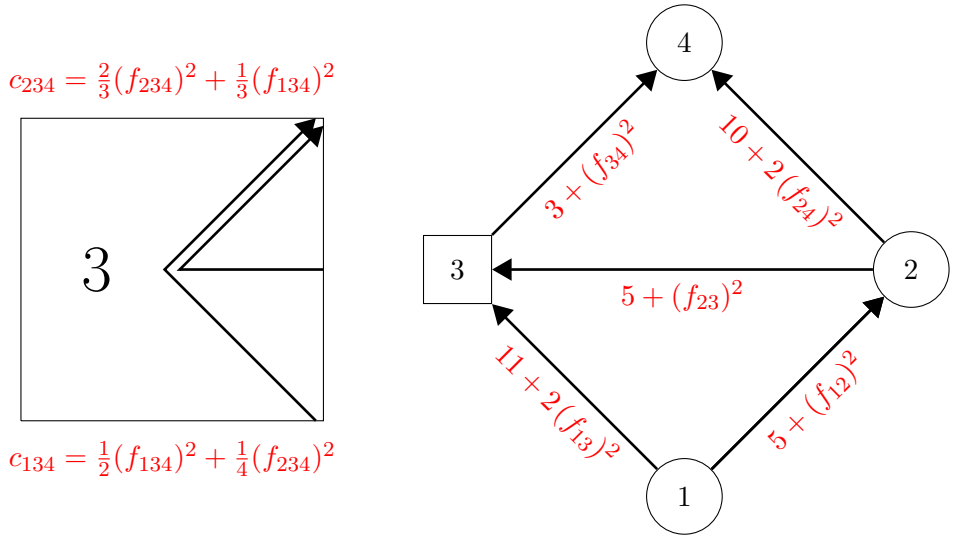
The number of travellers are real numbers instead of integers, so one may think that the flow must be an integer. Restricting the flow to integers makes the computing time unnecessarily high, since the demands in real life networks are very high and rounding does not affect the assignment significantly.

Before the example network is presented, we will look at the notation. The f stands for flow, but there are a lot of different flows: total flow through a link, flow through a path, flow with origin r, et cetera. The flow leaving node i has the notation f_i. The flow through a link can be subdivided by the total flow (f_{ij}), the flow with origin r ($f_{ij}^{r,\bullet}$), the flow with destination s ($f_{ij}^{\bullet,s}$) and the flow of OD-pair rs ($f_{ij}^{r,s}$). The cost of link ij is c_{ij}. There exists a junction in our example network. The flow from node i to node j via the junction of node $\tilde{i}$ is $f_{i\tilde{i}j}$ and has cost $c_{i\tilde{i}j}$ for using the junction.

Example network

The example network and interpretation of the cost function is described in this subsection.

We will use the network of Figure 4.0.1 as example network for the IA and MSA. All other algorithms will be done on that network without the junction (we assume node 3 to be a node without junction modelling), since we have to calculate the amount of flow we want to shift in these algorithms. Calculating the turn costs in a junction is often time intensive, due to the complicated tun cost function. The amount of flow we want to shift is dependent on the turn costs, so calculating the optimal shift is time intensive. We refer to Section 5.3.3 for more information about the turn costs.
The circles in Figure 4.0.1 are nodes, which can be an origin or destination, and the rectangle is a junction. A junction consists of multiple turns. The only possible turns at junction 3 are from node 1 to 4 and from node 2 to 4, since link 34 is the only link leaving node 3. The cost of a turn is dependent on the flow of other turns in the junction or on other links.



(a) Enlargement of node 3.

(b) The circles are nodes. The rectangle is a junction enlarged at (a).

Figure 4.0.1: The example network with demands $D_{1,3} = 2$, $D_{1,4} = 9$ and $D_{2,4} = 2$ travellers. The red formula gives the cost function of that link. If an algorithm cannot handle junctions we assume node 3 ordinary, if an algorithm can handle junctions we assume node 3 to be the junction described in (a)

The demands are given in the following table:

OD-pair	Demand
13	2
14	9
24	2

Now that the example network is described we will look at the frequently used AON-assignment and perform an AON-assignment on this example network.

All-or-nothing-assignment (AON-assignment)

Multiple algorithms in this thesis begin with an AON-assignment, therefore we will do an AON-assignment on the example network in this subsection.
The AON-assignment is an assignment where the demand of an OD-pair is put entirely on the minimum cost path. Only one path is selected when there are multiple minimum cost paths. In this way a link or path has either all the flow of an OD-pair or none.

We begin the AON-assignment with determining the cost of every link and turn based on the current flow. In our case there is no initial solution, so there is no flow at the network. The cost of a link or turn with zero flow is calculated. For link 12 we have $c_{12}(0) = 5+0^2 = 5$. The zero flow costs of all links and corresponding minimum cost paths can be found in Figure 4.0.2.

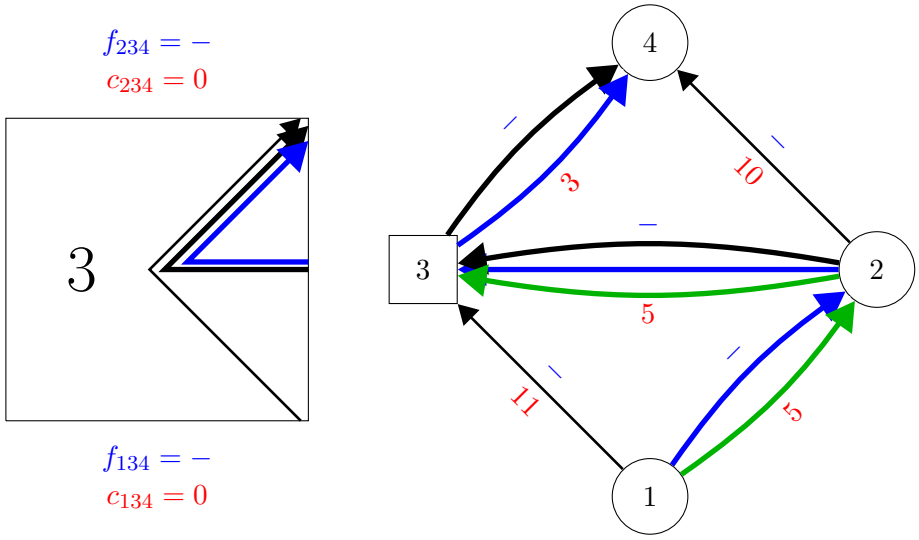


Figure 4.0.2: The example network with zero flow. The blue "-" above a link means there is no flow on that link. The red number below a link is the cost of the link based on zero flow. The green thick arrows mark the minimum cost path of OD-pair 13*, the blue ones of* 14 *and the thick black arrows mark the minimum cost path of OD-pair* 24*.*

The assignment of the demands to these minimum cost paths can be found in Figure 4.0.3, where the green number represents the flow of OD-pair 13, the blue number of 14 and the black number of 24. The red number below a link states the cost of that link. This cost is calculated after the assignment is done. In general, the cost of a link in a figure is the cost belonging to the flow of that link in that figure.

To get familiar with the notation we give Figure 4.0.3 in formula:

$f_{12}^{1,\bullet} = 11 \quad f_{13}^{1,\bullet} = 0 \quad f_{23}^{1,\bullet} = 11 \quad f_{24}^{1,\bullet} = 0 \quad f_{34}^{1,\bullet} = 9 \quad f_{134}^{1,\bullet} = 0 \quad f_{234}^{1,\bullet} = 9$
$f_{12}^{2,\bullet} = 0 \quad f_{13}^{2,\bullet} = 0 \quad f_{23}^{2,\bullet} = 2 \quad f_{24}^{2,\bullet} = 0 \quad f_{34}^{2,\bullet} = 2 \quad f_{134}^{2,\bullet} = 0 \quad f_{234}^{2,\bullet} = 2$

$f_{12}^{\bullet,3} = 2 \quad f_{13}^{\bullet,3} = 0 \quad f_{23}^{\bullet,3} = 2 \quad f_{24}^{\bullet,3} = 0 \quad f_{34}^{\bullet,3} = 0 \quad f_{134}^{\bullet,3} = 0 \quad f_{234}^{\bullet,3} = 0$
$f_{12}^{\bullet,4} = 9 \quad f_{13}^{\bullet,4} = 0 \quad f_{23}^{\bullet,4} = 11 \quad f_{24}^{\bullet,4} = 0 \quad f_{34}^{\bullet,4} = 11 \quad f_{134}^{\bullet,4} = 0 \quad f_{234}^{\bullet,4} = 11$

$f_{12} = 11 \quad f_{13} = 0 \quad f_{23} = 13 \quad f_{24} = 0 \quad f_{34} = 11 \quad f_{134} = 0 \quad f_{234} = 11$

$f_1 = 11 \quad f_2 = 13 \quad f_3 = 11 \quad f_4 = 0$

We can see in that the flows are updated, but the costs are dependent on these flows. When a flow changes, we have to update the costs.
In Section 3.1 we mentioned origin-based as example of a bush-based assignment. The flows $f_{ij}^{1,\bullet}$ (the flow with origin 1) and $f_{ij}^{2,\bullet}$ are stored when the algorithm is origin-based. The bush of one origin and one destination are determined in the next subsection.

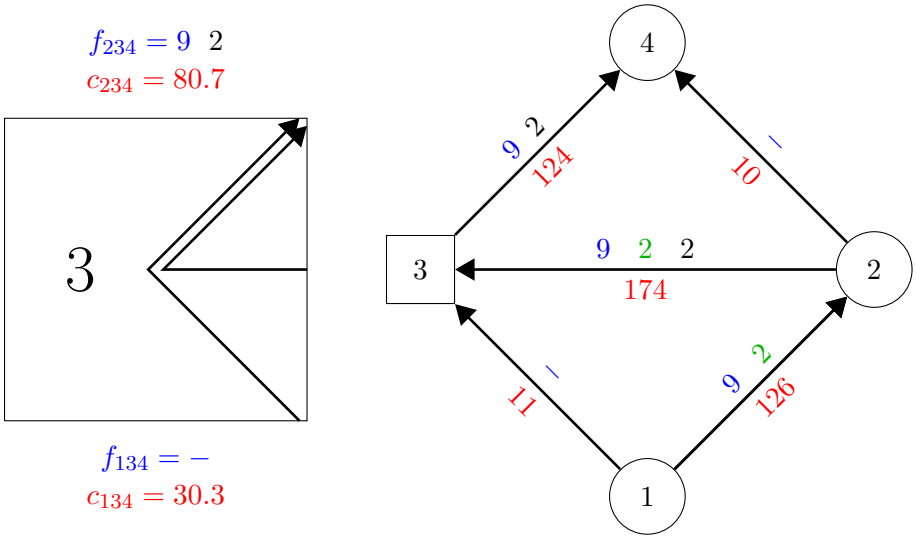


Figure 4.0.3: An AON-assignment on our example network. The green number represents the flow of OD-pair 13*, the blue number of* 14 *and the black number of* 24*. The red number states the cost of a link.*

Bush and bush modification

This subsection explains the need of bush modification and determines the bush and its improvement of origin 2 and destination 4.

A bush-based algorithm stores the flow per bush every iteration, where a bush consists of all path of a group of OD-pairs. Suppose we have a destination-based algorithm, thus a group is all OD-pairs of one destination. A bush consists of all links used by destination s. The modified bush consists of the bush plus some paths with less cost then the current minimum cost path used by an OD-pair. One can improve the bush by adding only the cheapest path, all paths with less cost or something in between. But why would we modify the bush? In equilibrium we cannot modify the bush, since all used paths between an OD-pair have minimal cost. During iterations the flow within a bush is shifted to minimize the cost. If we do not modify the bush, no links are added and the origin cannot use other links than it had during the initial solution.
We need to know which paths have less costs then the cheapest path in the bush to perform a bush modification. How these paths are found is explained here. Suppose we want the bush of a destination. This can be done by comparing the cost of the minimum cost path from a node to the destination, C_i^s, with the cost of the minimum cost path from another node, C_j^s. We add a link ij if $C_i^s \geq C_j^s + c_{ij}$ and drop a link when there is no flow on that link. The improvement of the bush consists of finding all nodes with the property $C_i^s \geq C_j^s + c_{ij}$, given there is a path $r - ... - i - j - ...s$ and demands $D_{r,s} > 0$, and finding all links in the bush with zero flow.
The origin-bush is determined analogously. We will calculate first the bush of destination 4 and thereafter the bush of origin 2.

The bush of destination 4 is equal to the used links, so it consists of the links 12, 23 and 34. To determine the modification of this bush we have to calculate the minimum cost paths in the network from any node to destination 4.
$C_4^4 = 0$,
$C_3^4 = \min\{124\} = 124$,
$C_2^4 = \min\{174 + 80.7 + 124, 10\} = \min\{378.7, 10\} = 10$ and
$C_1^4 = \min\{126 + 174 + 80.7 + 124, 126 + 10, 11 + 30.3 + 124\} = \min\{504.7, 136, 165.3\} = 136$.
We already have flow on links 12, 23 and 34, so that is the bush. To determine which links ij we have to add, we have to look for $C_i^s \geq C_j^s + c_{ij}$. We can see that $C_2^4 = 10 \geq 10 = C_4^4 + c_{24}$, so we have to add link 24. This is not the case for link 13, thus we don't add this link to the modified bush. The modified bush of destination 4 is the entire network except link 13. These results can be found in Figure 4.0.4, where besides the minimum path cost in the network also the maximum cost in the bush is stated. This maximum cost is added to show the difference between the used paths and the minimum paths.

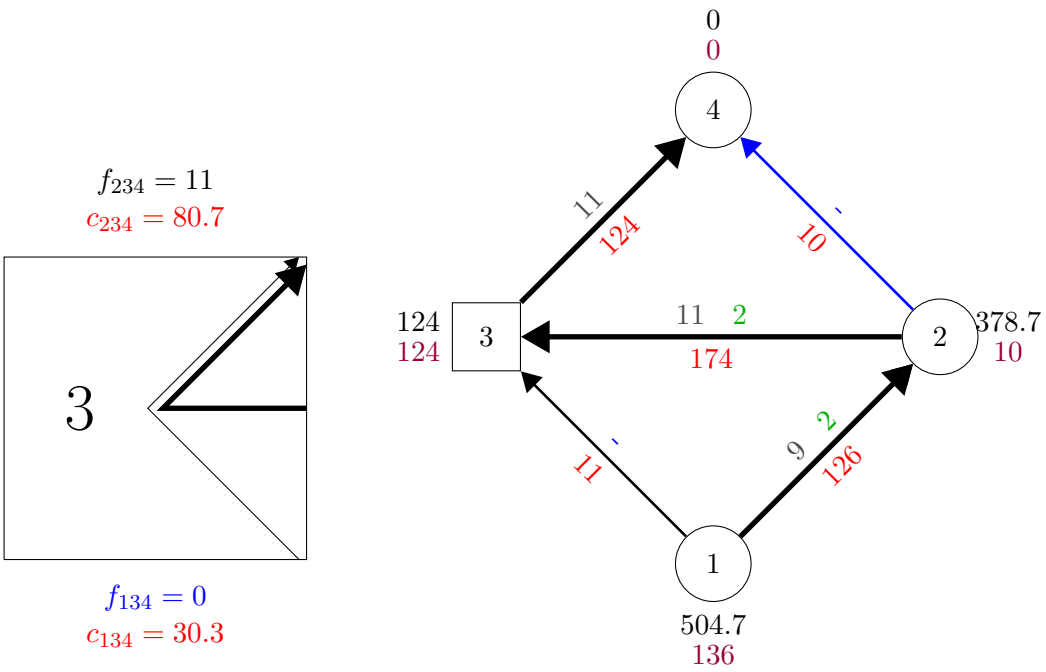


Figure 4.0.4: The (modified) bush of destination 4 based on Figure 4.0.3. The thick arrows mark the bush of destination 4. The blue lines are added during bush modification. The green number above a link is the flow with destination 3 and the grey number is the flow with destination 4. The upper number next to a node is the maximum cost used by destination 4 and the lower number is the cost of a minimum path from that node to destination 4.

The (modified) bush of origin 2 is calculated likewise. This bush is also based on Figure 4.0.3. We identify the bush by detecting the flow on 23 and 34. For the modification we need the cost from origin 2 to any other node, which can be found in Figure 4.0.5. Here we see that $C_2^4 = 10 \geq 10 = C_2^2 + c_{24}$. The modified bush exists of the links 23, 34 and 24.

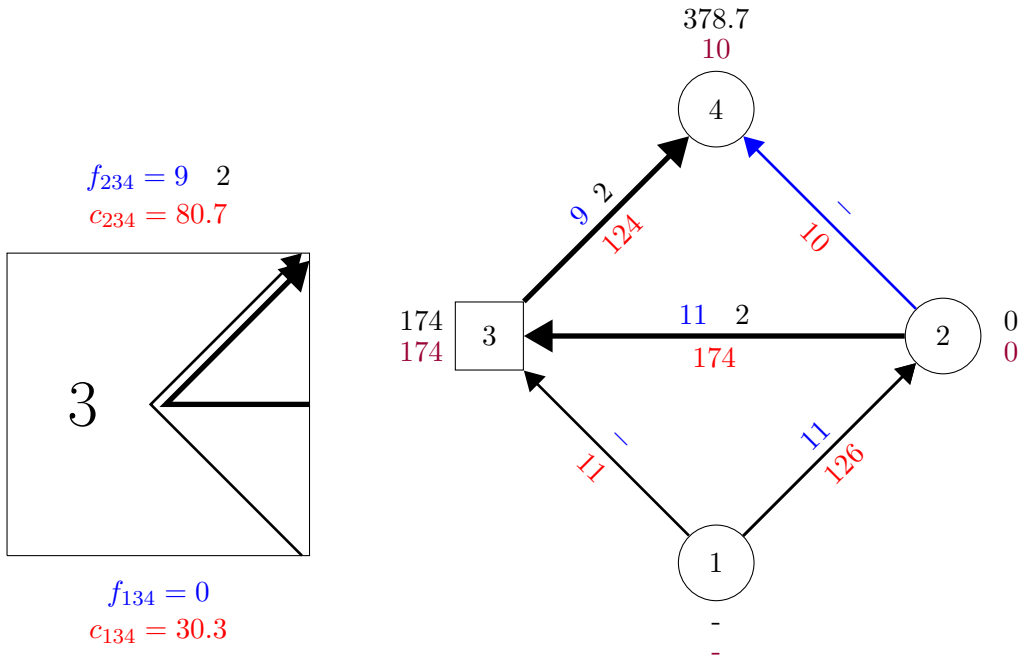


Figure 4.0.5: The (modified) bush of origin 2 based on Figure 4.0.3. The thick arrows mark the bush of origin 2. The blue lines are added during bush modification. The blue number above a link is the flow with origin 1 and the black number is the flow with origin 2. The upper number next to a node is the maximum cost used by destination 2 and the lower number is the cost of a minimum path from origin 2 to that node, where "–" means no such path exists.

4.1 Incremental assignment (IA)

In this section we will describe the incremental assignment (IA).
Most programmes, including OmniTRANS, store link-flows when using this algorithm. It is possible to store path-flows or bush-flows, but in the example and in the property table we will assume this algorithms is link-based.
The idea of this algorithm is assigning a fraction of the demand every iteration. OmniTRANS is able to use junction modelling when applying the incremental assignment, therefore we do this algorithm on a network with junction modelling.

Description

A predetermined fraction of all demands is assigned according to an AON-assignment based on the cost of the previous iteration. One can choose a different fraction for every iteration. For example, one can choose ten iterations with a fraction of 0.1 or one can choose $0.4, 0.3, 0.2$ and 0.1 for respectively the first, second third and fourth iteration.
In formula we have

$$\mathbf{f}^{(n)} = \mathbf{f}^{(n-1)} + \lambda_n \, \mathbf{e}^{(n)}$$

where $\mathbf{f}^{(n-1)}$ is the flow of iteration $n-1$ and $\mathbf{e}^{(n)}$ is the AON-assignment of iteration n.

Example

To make the algorithm more clear we will give an example on the network of Figure 4.0.1. The fractions used are $0.5, 0.3$ and 0.2 for respectively the first, second and third iteration.

Every iteration starts with determining an AON-assignment based on the current costs. This is already done at the beginning of this chapter and can be found in Figure 4.0.3. The fraction is 0.5 for the first iteration, thus $0.5 \cdot D_{r,s}$ is assigned to the minimum cost path of OD-pair rs. We have $f_{12}^{(1)} = 0.5 \cdot (2+9) = 5.5$, $f_{23}^{(1)} = 0.5 \cdot (9+2+2) = 6.5$ and $f_{34}^{(1)} = 0.5 \cdot (9+2) = 5.5$ travellers. This assignment together with the matching costs can be found in Figure 4.1.1. Iteration 2 starts with determining the minimum cost paths hereon which are marked by thick arrows in this figure.

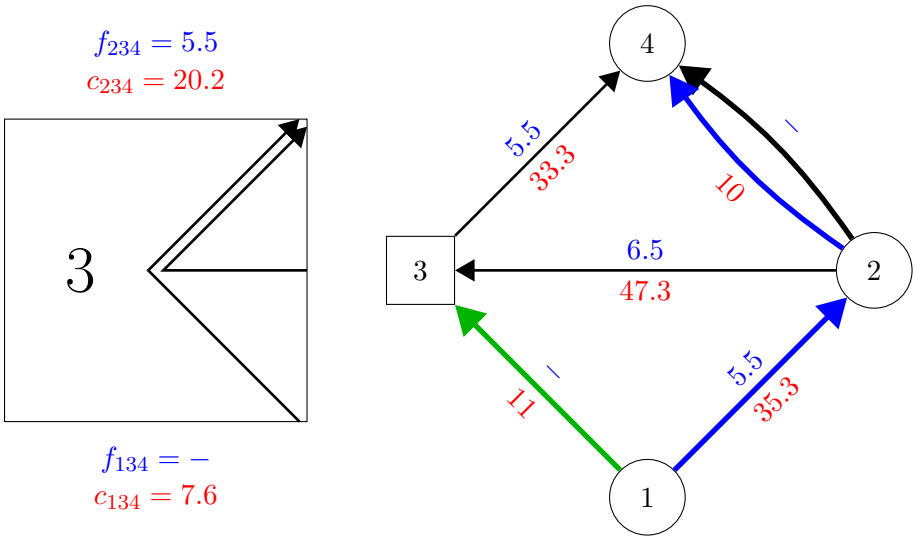


Figure 4.1.1: Assignment after 1 iteration of IA. The blue number represents the flow and the red number the cost of a link. The thick green arrow indicates the minimum cost path for OD-pair 13*, the blue ones for* 14 *and the thick black one for* 24*.*

The next step of iteration 2 is summing up the assignment of iteration 1 and 0.3 times the new AON-assignment. The new AON-assignment is sending 2 flow through the thick green arc, 9 through the thick blue arcs and 2 through the thick black arcs of Figure 4.1.1. We will calculate a few links as illustration: $f_{24}^{(2)} = f_{24}^{(1)} + 0.3 \cdot e_{24}^{(2)} = 0 + 0.3 \cdot (9+2) = 3.3$, $f_{34}^{(2)} = 5.5 + 0.3 \cdot 0 = 5.5 + 0 = 5.5$ and $f_{12}^{(2)} = 5.5 + 0.3 \cdot 9 = 8.2$. Iteration 2 results in Figure 4.1.2.

For iteration 3 we begin with determining the minimum cost paths based on the costs of iteration 2. The minimum cost paths are $1-3$, $1-2-4$ and $2-4$. The fraction of flow we must assign is 0.2. We add $f_{13}^{(3)} = 0.2 \cdot 2 = 0.4$, $f_{12}^{(3)} = 0.2 \cdot 9 = 1.8$ and $f_{24}^{(3)} = 0.2 \cdot 11 = 2.2$ travellers to the previous iteration, which gives Figure 4.1.3.

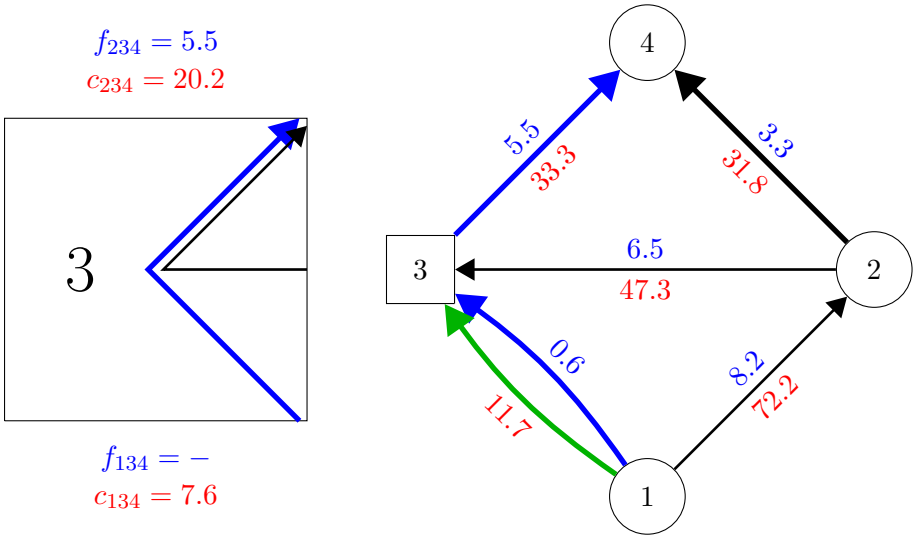


Figure 4.1.2: Assignment after 2 iterations of IA. The blue number represents the flow and the red number the cost of a link. The thick green arrow indicates the minimum cost path for OD-pair 13*, the blue ones for* 14 *and the thick black one for* 24*.*

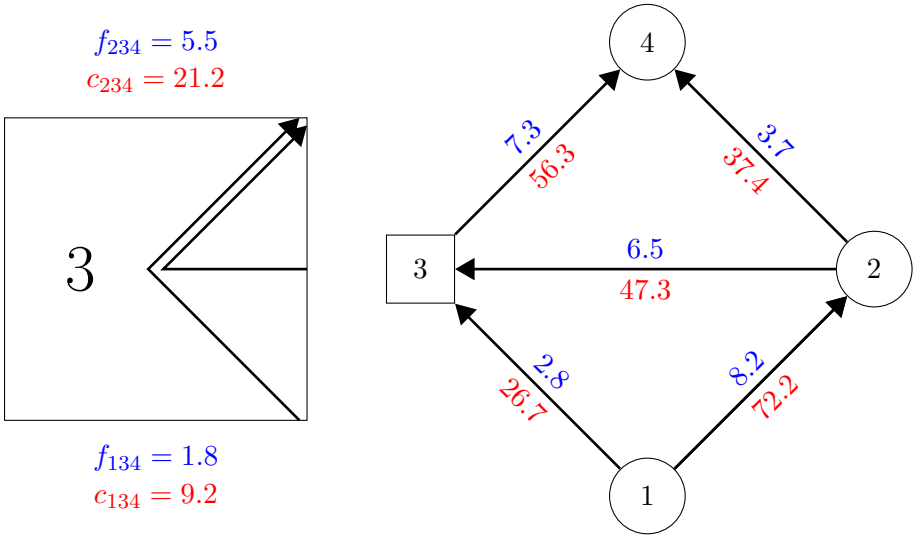


Figure 4.1.3: Assignment after 3 iterations of IA. The blue number above a link represents the flow and the red number the cost of a link.

Properties

Memory usage	++
Proportionality	−
Rate of convergence	n/a

4.2 Method of successive averages (MSA)

MSA is a specification of volume averaging (VA). All demand is assigned in the first iteration by an AON-assignment. During every iteration a predetermined ratio of the last iteration, $1-\lambda_n$, and the AON-assignment, λ_n, based on this last iteration is taken as new assignment. This algorithm is called VA in general and MSA if λ_n is specified as $\lambda_n = 1/n$. OmniTRANS is able to use junction modelling when applying MSA, therefore we do this algorithm on a network with junction modelling.

Description

Every iteration begins with searching an AON-assignment based on the cost of the previous iteration. For the first iteration an AON-assignment is done based on the free-flow costs. The new assignment consists of a proportion of the previous assignment and a proportion of the last found AON-assignment. In formula we have $(1-\lambda_n)\mathbf{f}^{(n-1)}+\lambda_n\mathbf{e}^{(n)} = \mathbf{f}^{(n-1)}+\lambda_n\cdot(\mathbf{e}^{(n)}-\mathbf{f}^{(n-1)})$, where $\mathbf{e}^{(n)}$, the alternative flow at iteration n, is the AON-assignment.

Example

This section gives 3 iterations of MSA on the example network. We have $\lambda_n = 1/n$. The first iteration consists of finding an AON-assignment. This is already done at the beginning of this chapter and can be found in Figure 4.0.3. The second iteration is finding an AON-assignment based on these costs. The minimum cost paths are $1-3$, $1-2-4$ and $2-4$. The corresponding assignment and costs can be found in Figure 4.2.1.

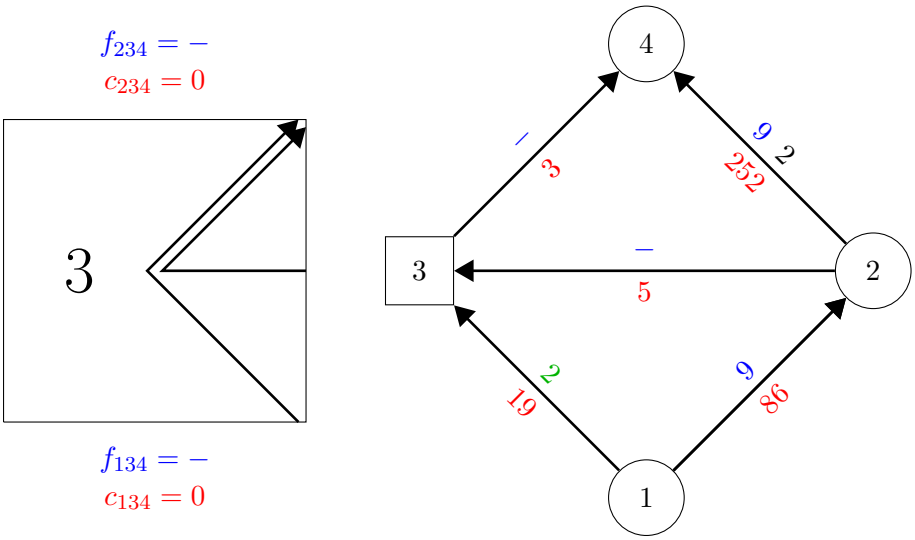


Figure 4.2.1: The AON-assignment based on the assignment of iteration 1 of MSA (Figure 4.0.3). The green number is the flow of OD-pair 13*, blue of* 14 *and black of* 24*. The red number is the costs of a link.*

We have $\lambda_n = 1/n$, thus for iteration 2 we have $\lambda_2 = 1/2$. Taking half of iteration 1 (Figure 4.0.3) and half of the AON-assignment based on the costs of iteration 1 (Figure 4.2.1)

gives Figure 4.2.2.

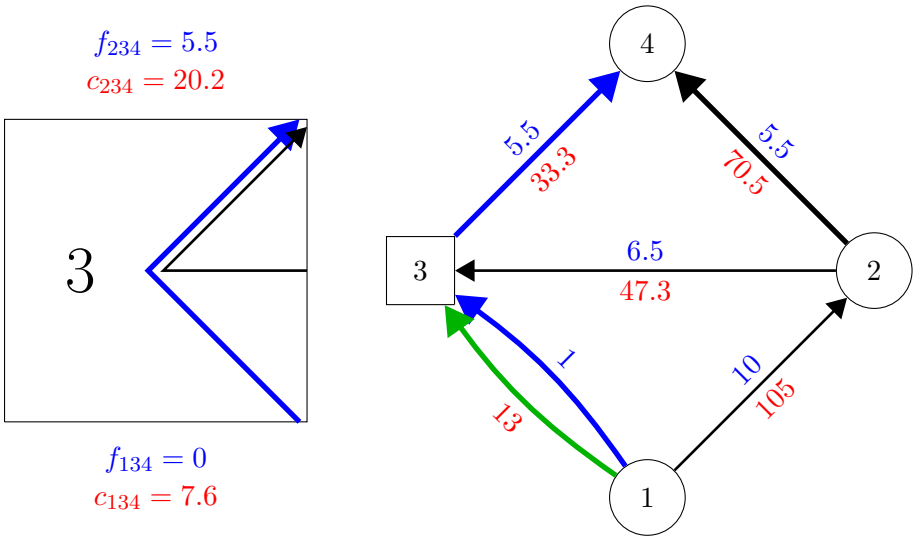


Figure 4.2.2: Assignment after 2 iterations of MSA. The blue number represents the flow and the red number the cost of a link. The thick green arrow indicates the minimum cost path for OD-pair 13*, the blue ones for* 14 *and the thick black one for* 24*.*

The minimum cost paths in this assignment are $1-3$ for OD-pair 13, $1-3-4$ for OD-pair 14 and $2-4$ for pair 24. Since we are working on iteration 3 we have $\lambda_3 = 1/3$. Doing $2/3 \cdot \mathbf{f}^{(2)} + 1/3 \cdot \mathbf{e}^{(3)}$ gives Figure 4.2.3.

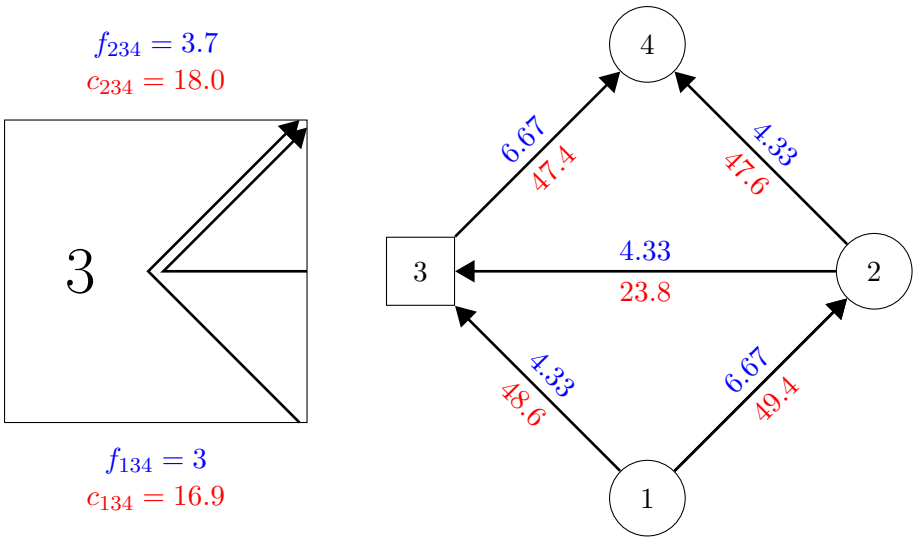


Figure 4.2.3: Assignment after 3 iterations of MSA. The blue number is the flow and the red number the cost of a link.

Properties

Memory usage	++
Proportionality	+
Rate of convergence	−

4.3 Frank-Wolfe (FW)

The Frank-Wolfe (FW) method is published by M. Frank and P. Wolfe in 1956 [13]. There are many variants and extensions based on this algorithm. FW looks like MSA, but instead of a predetermined fraction between the old assignment and the new AON-assignment FW uses a line search to determine the optimal fraction. We will use the network without junction modelling for FW and the next algorithms, since we have to determine the fraction ourselves and this may be time intensive, depending on the turn cost function. This is explained into greater detail in section 5.3.3.

Description

One makes an AON-assignment based on the current link costs. Then a λ is chosen such that $O(\mathbf{f}^{(n-1)} + \lambda(\mathbf{e}^{(n)} - \mathbf{f}^{(n-1)})) = O((1-\lambda)\mathbf{f}^{(n-1)} + \lambda\mathbf{e}^{(n)})$ is minimized.

Example

We will show three iterations of the FW algorithm in this subsection.

The first iteration consists of finding an AON-assignment on the zero-flow network. The second iteration begins with finding an AON-assignment based on the costs of the first iteration. This is exactly the beginning of MSA. The first iteration is the same as Figure 4.0.3, except that we have no junction modelling for FW. The shortest paths used for the AON-assignment based on the first iteration are $1-3$, $1-3-4$ and $2-4$. This AON-assignment can be found in Figure 4.2.1. Instead of using a predetermined fraction, as in MSA, we want to

$$\min_{0\leq\lambda\leq 1} \int_0^{11+\lambda(0-11)} c_{12}(w)\,\mathrm{d}w + \int_0^{0+\lambda(11-0)} c_{13}(w)\,\mathrm{d}w + \int_0^{13+\lambda(0-13)} c_{23}(w)\,\mathrm{d}w + \int_0^{0+\lambda(2-0)} c_{24}(w)\,\mathrm{d}w + \int_0^{11+\lambda(9-11)} c_{34}(w)\,\mathrm{d}w$$

The λ fulfilling this criterion is 0.56. The assignment $\mathbf{f}^{(2)} = \mathbf{f}^{(1)} + 0.56 \cdot (\mathbf{e}^{(2)} - \mathbf{f}^{(1)})$ can be found in Figure 4.3.1.

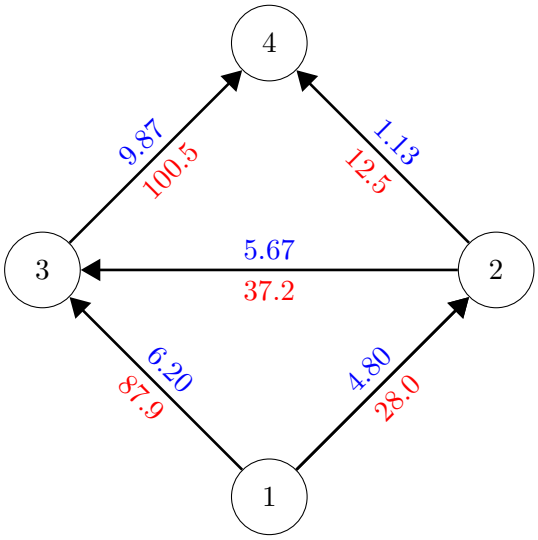


Figure 4.3.1: Assignment after 2 iterations of FW. The blue number is the flow and the red number the cost of a link.

Now the second iteration is calculated we are ready for the third iteration. Again an AON-assignment is made based on the previous iteration, iteration 2. The minimum paths are $1-2-3$ for OD-pair 13, $1-2-4$ for OD-pair 14 and $2-4$ for OD-pair 24. Minimizing the objective function with a combination of the AON-assignment and the assignment of iteration 2 ($\min_{0\leq\lambda\leq1} O(\mathbf{f}^{(2)} + \lambda\,(\mathbf{e}^{(3)} - \mathbf{f}^{(2)})$) gives $\lambda = 0.19$ and the assignment of Figure 4.3.2.

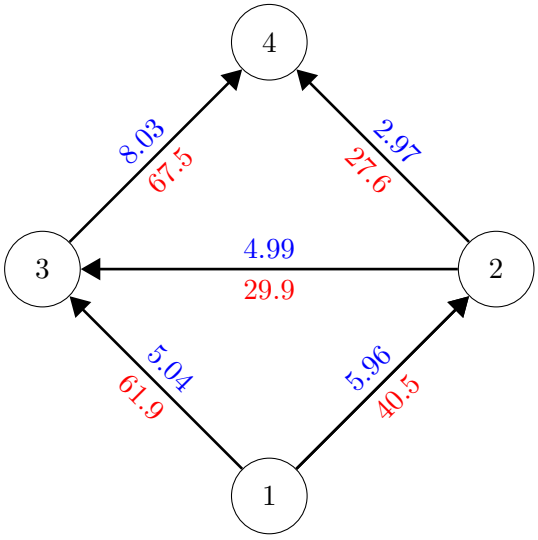


Figure 4.3.2: Assignment after 3 iterations of FW. The blue number is the flow and the red number is the cost of a link.

Properties

Memory usage	++
Proportionality	+
Rate of convergence	−

One way to handle junctions is calculating the cost of every turn at the end of every iteration. One can take this function as a constant for the next iteration.

4.4 Simplicial decomposition (SD)

Simplicial decomposition (SD) is FW in multiple dimensions.

The FW method stores the flow per link of the last iteration. An extension to FW is storing multiple iterations, say N, and using these to determine the flows in a new iteration.
The fraction of assignments one uses can be a constant. For example, $\mathbf{f}^{(n+1)} = \frac{3}{6}\mathbf{f}^{(n)} + \frac{2}{6}\mathbf{f}^{(n-1)} + \frac{1}{6}\mathbf{f}^{(n-2)}$. When one extra iteration is stored ($N = 2$), the procedure is called Conjugated Frank Wolfe. In case of storing two extra iterations ($N = 3$), the algorithm is called Bi-Conjugated Frank Wolfe.
Another possibility is not to use a constant, but to determine the optimal fraction every iteration. This is done in SD.

Description

FW stores only the last iteration where SD stores multiple iterations. The number of iterations stored, N, is determined before the algorithm starts. An iteration begins with making an AON-assignment based on the costs of the previous iteration. The new assignment is a linear combination of the last N iterations and the AON-assignment which minimizes the objective function. In formula we have

$$\min_{\substack{0\leq\lambda_1,\ldots,\lambda_N\leq 1\\ \sum_{\tilde{n}=1}^{N}\lambda_n\leq 1}} O\Big(\sum_{\tilde{n}=n-N+1}^{n-1} \lambda_{\tilde{n}}\,\mathbf{f}^{(\tilde{n})} + \lambda_n\,\mathbf{e}^{(n)}\Big)$$

Example

The first iteration is the same as FW: finding an AON-assignment based on the free flows. The second iteration is also the same, since we now have a combination of the first two assignments. The third iteration begins the same, namely searching for the same AON-assignment based on iteration 2. From here the algorithms differ. FW searched for a combination of $\mathbf{f}^{(2)}$ and $\mathbf{e}^{(3)}$ where SD searches for a combination of $\mathbf{f}^{(1)}$, $\mathbf{f}^{(2)}$ and $\mathbf{e}^{(3)}$. So we want to minimize $\min_{\substack{0\leq\lambda_1,\lambda_2\leq 1\\ \lambda_1+\lambda_2\leq 1}} O(\lambda_1\,\mathbf{f}^{(1)} + \lambda_2\,\mathbf{f}^{(2)} + (1-(\lambda_1+\lambda_2))\,\mathbf{e}^{(3)}$. When these λ_1 and λ_2 are found one can construct the new assignment and go on with the next iteration.

Properties

Memory usage	+
Proportionality	+
Rate of convergence	+−

4.5 Projected gradient method (PG)

The basis of this section is [3]. Since there are some mistakes in this paper we have explained in the appendix what we changed and used our revised version.
This subsection first gives an idea of the algorithm in "description" after which the algorithm is given in formula. Next an example is given and we end this section with an overview of the properties.

Description

The idea of this algorithm is to look at an OD-pair, lower the cost of active paths and go to the next OD-pair. A set of active paths consists of the used paths and the paths we want to use (minimum cost paths). For an OD-pair one determines the difference for every active path between the average cost of the active paths and the cost of a path. Every active path is updated by a factor times this difference, where the factor minimizes the objective function. Next one searches for a new minimum cost path. If there is no such path the next OD-pair is considered, if there is such a path this path is added to the active paths and the OD-pair is evaluated again with the new set of active paths.
In formula we have for every OD-pair
Step 0 Compute an initial solution and determine $P_{r,s}^+ = \{p | f_{\{p\}}^{r,s} > 0\}$.

Step 1 Compute the descent direction for every active path:

$$g_{\{p\}} = \bar{c}_{r,s} - c_{\{p\}} \quad \forall p \in P_{r,s}^+.$$

where $\bar{c}_{r,s}$ is the average cost of a path of OD-pair rs.
If $\max_{p \in P_{r,s}^+} (|g_{\{p\}}|) < \epsilon$ go to *Step 4*.

Step 2 Find the optimal step size λ^* which is the solution of the subproblem

$$\min_{\lambda} \sum_{ij \in A} \int_0^{f_{ij}^{r,s} + \bar{f}_{ij}^{r,s} + \lambda G_{ij}^{r,s}} c_{ij}(w) \mathrm{d}w$$

with the restriction

$$0 \leq \lambda \leq \min \left\{ \frac{-f_{\{p\}}}{g_{\{p\}}} \middle| g_{\{p\}} < 0 \right\}$$

where

$$G_{ij}^{r,s} = \sum_{\{p\} \in P_{r,s}^+} \delta_{ij}^{\{p\}} g_{\{p\}}$$

and

$$\bar{f}_{ij}^{r,s} = \sum_{\substack{(r,s) \in D \\ r \neq s}} \sum_{p \in P_{r,s}^+} \delta_{ij}^{\{p\}} f_{\{p\}}$$

is the fixed flow of all other OD-pairs of the network.

Step 3 Update the path flows and link flows

$$f_{\{p\}} = f_{\{p\}} + \lambda^* g_{\{p\}}, \;\; p \in P^+_{r,s}$$
$$f^{r,s}_{ij} = f^{r,s}_{ij} + \lambda^* G^{r,s}_{ij}, \;\; ij \in A$$

If a path $\hat{p}$ diminishes to zero ($f_{\hat{p}} \to 0$) it is eliminated, that is, $P^+_{r,s} = P^+_{r,s} - \hat{p}$.

Step4 Compute the minimum cost path $\tilde{p}$ of minimal costs $c_{\{\tilde{p}\}}(\mathbf{f}) = \min_{p \in P^+_{r,s}} c_{\{p\}}(\mathbf{f})$.
If $c_{\{\tilde{p}\}}(\mathbf{f}) \leq \min_{p \in P^+_{r,s}} c_{\{p\}}(\mathbf{f})$, then add path $\tilde{p}$ to the set of active paths $P^+_{r,s}$ ($P^+_{r,s} = P^+_{r,s} + \tilde{p}$) and return to *Step 1.* Otherwise, stop.

Example

We assume $\epsilon = 0.01$. The initial solution is the AON-assignment calculated in the beginning of this chapter, which is Figure 4.0.3. From this figure we get $P^+_{13} = \{1-2-3\}$, $P^+_{14} = \{1-2-3-4\}$ and $P^+_{24} = \{2-3-4\}$.

We start with OD-pair 13.

Step 1
The descent directions are:
$c_{\{1-2-3\}} = 126 + 174 = 300$
$\bar{c}_{1,3} = c_{\{1-2-3\}} = 300$
$g_{\{1-2-3\}} = \bar{c}_{1,3} - c_{\{1-2-3\}} = 300 - 300 = 0$
$\max_{p \in P^+_{13}}\{|g_{\{p\}}|\} = \max\{|g_{\{1-2-3\}}|\} = 0 < \epsilon \Rightarrow$*Step 4*

Step 4
The minimum cost path between origin 1 and destination 3 is $1-3$. We have to add this path to P^+_{13}, so we get $P^+_{13} = \{1-2-3, 1-3\}$. We have added a path, so we go back to *Step 1* to equilibrate the cost in the new set of active paths.

Step 1
$c_{\{1-2-3\}} = 300$
$c_{\{1-3\}} = 11$
$\bar{c}_{1,3} = \frac{1}{2}(300 + 11) = 155.5$
$g_{\{1-2-3\}} = \bar{c}_{1,3} - c_{\{1-2-3\}} = 155.5 - 300 = -144.5$
$g_{\{1-3\}} = \bar{c}_{1,3} - c_{\{1-3\}} = 155.5 - 11 = 144.5$
$\max_{p \in P^+_{13}}\{|g_{\{p\}}|\} = \max\{|-144.5|, |144.5|\} = 144.5 \not< \epsilon$, thus we go to *Step 2.*

Step 2
$\min\left\{ \frac{-f_{\{p\}}}{g_{\{p\}}} \middle| \, g_{\{p\}} < 0 \right\} = \min\{\frac{-2}{-144.5}\} = \frac{1}{72.25}$. This is the upper bound for λ. We are searching for a λ^* which satisfies:

$$\min_{0\leq\lambda\leq 1/72.25} \int_0^{2+9+\lambda\cdot(-144.5)} c_{12}(w)\,\mathrm{d}w + \int_0^{0+0+\lambda\cdot(144.5)} c_{13}(w)\,\mathrm{d}w + \int_0^{2+11+\lambda\cdot(-144.5)} c_{23}(w)\,\mathrm{d}w + \int_0^{0+0+\lambda\cdot(0)} c_{24}(w)\,\mathrm{d}w + \int_0^{2+9+\lambda\cdot(144.5-144.5)} c_{34}(w)\,\mathrm{d}w$$

Solving this minimization problem gives $\lambda^* = \frac{1}{72.25}$ and $g_{\{1-2-3\}}\,\lambda^* = -2$. We can see this number is right, since path $1-2-3$ has only a flow of 2, thus we cannot change more, and in the next step we will see that changing 2 flow makes the network more equilibrated.

Step 3
Changing the flow with $f_{\{p\}} = f_{\{p\}} + \lambda^* g_{\{p\}}$ gives Figure 4.5.1.

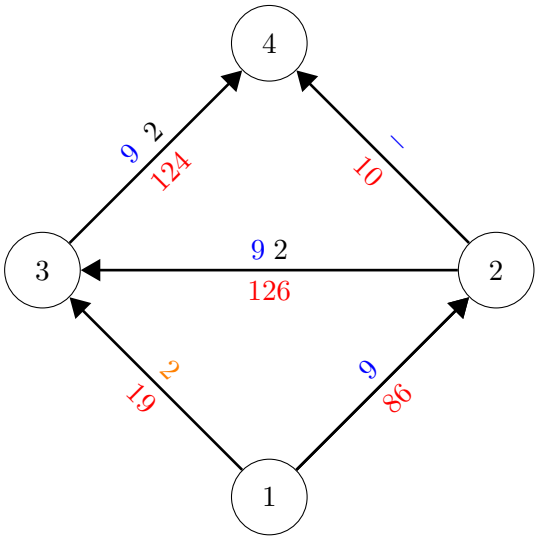


Figure 4.5.1: Assignment after 1 iteration of PG. The red number below a link is the cost of that link. The orange number is the flow of path $1-3$, blue of $1-2-3-4$ and black of $2-3-4$.

Step4
We can delete path $1-2-3$ from P_{13}^+ since there is no flow left on this path. We get $P_{13}^+ = \{1-2\}$.
There is no cheaper path then the one used right now. Since there is no path added we choose another OD-pair to equilibrate.
We choose now OD-pair 14.

Step 1
We have $P_{14}^+ = \{1-2-3-4\}$. We will calculate the cost of every active path.
$c_{\{1-2-3-4\}} = 86 + 126 + 124 = 336$
$\bar{c}_{1,4} = c_{\{1-2-3-4\}} = 336$
$\max_{p\in P_{14}^+}\{|g_{\{p\}}|\} = g_{\{1-2-3-4\}} = \bar{c}_{1,4} - c_{\{1-2-3-4\}} = 336 - 336 = 0 < \epsilon \Rightarrow$ *Step 4*

Step 4
Since there was only one path there was nothing to equilibrate. In this step we are going to search for a path we can add. The minimum cost path from 1 to 4 is path $1-2-4$. Adding this path to the set of active paths gives $P_{14}^{+} = \{1-2-3-4, 1-2-4\}$.

Step 1
The descent directions for the active paths are:
$c_{\{1-2-3-4\}} = 336$
$c_{\{1-2-4\}} = 86 + 10 = 96$
$\bar{c}_{1,4} = \frac{1}{2}(c_{\{1-2-3-4\}} + c_{\{1-2-4\}}) = \frac{1}{2}(336 + 96) = 216$
$g_{\{1-2-3-4\}} = 216 - 336 = -120$
$g_{\{1-2-4\}} = 216 - 96 = 120$
$\max_{p \in P_{14}^{+}}\{|g_{\{p\}}|\} = \max\{|-120|, |120|\} = 120$. There is enough flow to change, so we go to *Step 2*

Step 2
The upper bound of λ is $\min\{\frac{-f_{\{p\}}}{g_{\{p\}}} | g_{\{p\}} < 0\} = \min\{\frac{-9}{-120}\} = \frac{3}{40}$. We want to minimize

$$\min_{0 \le \lambda \le 3/40} \quad \int_0^{\lambda(120)} c_{24}(w)\,\mathrm{d}w + \int_0^{11+\lambda(-120)} c_{23}(w)\,\mathrm{d}w + \int_0^{11+\lambda(-120)} c_{34}(w)\,\mathrm{d}w + \int_0^{11+\lambda(-120)+\lambda(120)} c_{12}(w)\,\mathrm{d}w + \int_0^{0+\lambda(0)} c_{13}(w)\,\mathrm{d}w$$

This minimization gives $\lambda^* = 0.05$.

Step 3
The amount of flow we have to change is $\lambda^* g_{\{1-2-4\}} = 5.45$ and $\lambda^* g_{\{1-2-3-4\}} = -5.45$. Changing the flows results in Figure 4.5.2.

Step 4
The minimum cost path from 1 to 4 is path $1-3-4$. This path is not used right now, so we add it to the set of active paths $P_{14}^{+} = \{1-2-3-4, 1-2-4, 1-3-4\}$. We have added a path, so we go to *Step 1.*

Step 1
The descent directions of the active paths are:
$c_{\{1-2-3-4\}} = 86 + 35.8 + 33.8 = 155.5$
$c_{\{1-2-4\}} = 86 + 69.5 = 155.5$
$c_{\{1-3-4\}} = 19 + 33.8 = 52.8$
$\bar{c}_{1,4} = \frac{1}{3}(155.5 + 155.5 + 52.8) = 121.3$
$g_{\{1-2-3-4\}} = 121.3 - 155.5 = -34.3$
$g_{\{1-2-4\}} = 121.3 - 155.5 = -34.3$
$g_{\{1-3-4\}} = 121.3 - 52.8 = 68.5$
$\max_{p \in P_{14}^{+}}\{|g_{\{p\}}|\} = \max\{|155.5|, |155.5|, |68.5|\} = 155.5 \not< \epsilon$, so we go to *Step 2.*

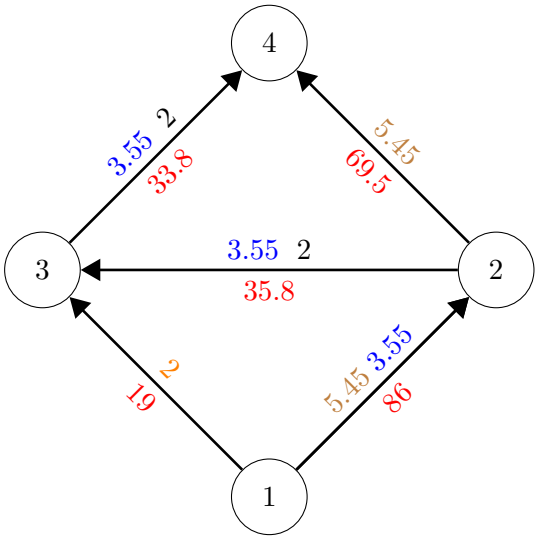


Figure 4.5.2: Assignment after 2 iterations of PG. The red number below a link is the cost of that link. The orange number is the flow of path $1-3$*, blue of* $1-2-3-4$*, black of* $2-3-4$ *and brown of* $1-2-4$*.*

Step 2
$\min\{\frac{-f_{\{p\}}}{g_{\{p\}}} | g_{\{p\}} < 0\} = \min\{\frac{-3.55}{-34.3}, \frac{-5.45}{-34.3}\} = \frac{3.55}{34.3}$. This is the upper bound for λ. We minimize the objective function with flow $f_{ij} = f_{ij}^{r,s} + \bar{f}_{ij}^{r,s} + \lambda G_{ij}^{r,s}$. This gives $\lambda^* = 0.05$ and as flow shift $\lambda^* g_{\{1-2-3-4\}} = -1.86$, $\lambda^* g_{\{1-3-4\}} = 3.72$ and $\lambda^* g_{\{1-2-4\}} = -1.86$.

Step 3
Changing the flow for all links with $f_{\{p\}} = f_{\{p\}} + \lambda^* g_{\{p\}}$ gives Figure 4.5.3.

Step 4
We use all possible paths, also the minimum cost path, so we cannot add a path. Since we do not add a path, we have to go to the next OD-pair. The only OD-pair we have not looked at yet is 24. We stop here with this example, but if we would go on with the next iterations we would go to OD-pair 24, next to 13, then to 14 et cetera.

Properties

Memory usage	$--$
Proportionality	$-$
Rate of convergence	$+$

4.6 Linear user cost equilibrium (LUCE)

This bush-based algorithm determines an alternative assignment for a bush and determines the optimal linear combination of the assignment of the last iteration and the alternative assignment. Our bush initialization differs from the paper. Why and how we changed it is explained in Appendix B.

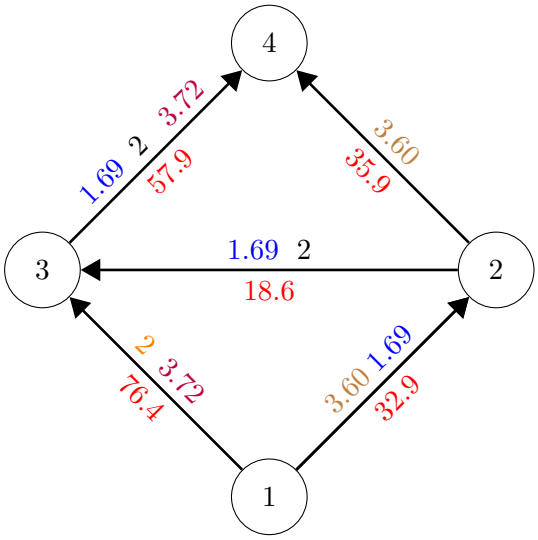


Figure 4.5.3: Assignment after 3 iterations of PG. The red number below a link is the cost of that link. The orange number is the flow of path $1-3$*, blue of* $1-2-3-4$*, black of* $2-3-4$*, brown of* $1-2-4$ *and purple of* $1-3-4$*.*

First we will give the main idea of the algorithm after which the pseudo code is given. Next the algorithm is provided with an example and the property table is given.

Description

The main idea of this algorithm is redistribute flow within a (modified) bush, where the amount of flow on a link is dependent on the cost and its derivative. LUCE assumes the link cost function is differentiable. The nodes are visited in reverse topological order to determine the average cost of the paths using node i, or the minimum path from i to the destination if i isn't used by any path, and its derivative. Visiting nodes in reverse topological order means starting at the destination node and visit nodes further and further away of the destination until every node is visited. The alternative assignment is determined by assigning flow to the nodes in topological order, where the cost and derivative play an important role. At the end the (approximated) optimal linear combination of the assignment of the last iteration and the alternative assignment is made.

The notation used in the pseudo code is N for the maximum number of iterations, f_{ij} for the flow on link ij and f_i^s is the flow from node i to destination s. The alternative flow on link ij is e_{ij} and e_i^s is the alternative flow from node i to destination s. The cost of a link with flow f_{ij} is $c_{ij}(f_{ij})$. The cost of a link which is determined at the beginning of an iteration has the notation c_{ij} and its derivative is g_{ij}. The average cost from node i to destination s denoted by $\tilde{C}_i^s$ and its derivative by $\tilde{G}_i^s$. y_{ij}^s is defined as 0 if $f_i^s = 0$. If this flow is greater than zero y_{ij}^s is defined as $f_{ij}^{\bullet,s}/f_i^s$. The modified or initialized bush $B(^s)$ is a function of the old bush, the cost and the flow of the network, notated by $B(B(^s), c^{(n)}, f^{(n)})$. V_i^s is a variable used for calculation simplification and J is the forward star bush $\mathrm{FSB}(i,s)$ of node i to destination s. An $\mathrm{FSB}(i,s)$ contains every node j such that $ij \in B(^s)$.
Now we'll give the pseudo code [4]:

$f^{(0)} = 0$
for $n = 1$ to N
  for each $s \in S$
    for each $ij \in A$
      $c_{ij} = \max\{c_{ij}(f_{ij}), \epsilon\}$
      $g_{ij} = \max\{\partial c_{ij}(f_{ij}), \epsilon\}$
      if $f_i^s > 0$ then $y_{ij}^s = f_{ij}^{\bullet,s}/f_i^s$ else $y_{ij}^s = 0$
    $B(^s) = B(B(^s), c^{(n)}, f^{(n)})$
    $\tilde{C}_s^s = 0$
    $\tilde{G}_s^s = 0$
    for each $i : \exists ij \in B(^s)$ in reverse topological order
      if $f_i^s > 0$ then
        $\tilde{C}_i^s = \sum_{j \in \mathrm{FSB}(i,s)} y_{ij}^s \cdot (c_{ij} + \tilde{C}_j^s)$
        $\tilde{G}_i^s = \sum_{j \in \mathrm{FSB}(i,s)} (y_{ij}^s)^2 \cdot (g_{ij} + \tilde{G}_j^s)$
      else
        $\tilde{C}_i^s = \min_{j \in \mathrm{FSB}(i,s)}\{c_{ij} + \tilde{C}_j^s\}$
        $\delta_j = 1$ if $\tilde{C}_i^s = c_{ij} + \tilde{C}_j^s$ and 0 otherwise
        $\tilde{G}_i^s = \sum_{j \in \mathrm{FSB}(i,s)} [\delta_j (g_{ij} + \tilde{G}_j^s)] / \sum_{j \in \mathrm{FSB}(i,s)} [\delta_j]$
    $e^s = 0$
    for each $r \in R$
      $e_r^s = D_{r,s}$
    for each $i : \exists ij \in B(^s)$ in topological order
      $J = \mathrm{FSB}(i, s)$
      $\lambda = 0$
      until $\lambda = 1$ do
        $\lambda = 1$
        $V_i^s = (e_i^s + \sum_{j \in J} [(c_{ij} + \tilde{C}_j^s)/(g_{ij} + \tilde{G}_j^s)] / \sum_{j \in J} [1/(g_{ij} + \tilde{G}_j^s)]$
        for each $j \in J$
          $e_{ij}^{\bullet,s} = V_i^s/(g_{ij} + \tilde{G}_j^s) - (c_{ij} + \tilde{C}_j^s)/(g_{ij} + \tilde{G}_j^s) + e_i^s \cdot y_{ij}^s$
          if $e_{ij}^{\bullet,s} < 0$ then
            $e_{ij}^{\bullet,s} = 0$
            $J = J \backslash \{j\}$
            $\lambda = 0$
        for each $j \in J$
          $e_j^s = e_j^s + e_{ij}^{\bullet,s}$
      $\lambda = 1$
      if $n > 1$ then
        $\partial O(1)/\partial\lambda = \sum_{ij \in A} c_{ij}(f_{ij} + \lambda(e_{ij}^{\bullet,s} - f_{ij}^{\bullet,s})) \cdot (e_{ij}^{\bullet,s} - f_{ij}^{\bullet,s})$
        if $\partial O(1)/\partial\lambda \leq 0$ then
          $\partial O(0)/\partial\lambda = \sum_{ij \in A} c_{ij} \cdot (e_{ij}^s - f_{ij}^{\bullet,s})$
          if $\partial O(0)/\partial\lambda \leq 0$ then $\lambda = 0$ else $\lambda = 1/(1 - (\partial O(1)/\partial\lambda) \,/\, (\partial O(0)/\partial\lambda))$
      for each $ij \in A$
        $f_{ij} = f_{ij} + \lambda \cdot (e_{ij}^{\bullet,s} - f_{ij}^{\bullet,s})$
        $f_{ij}^{\bullet,s} = f_{ij}^{\bullet,s} + \lambda \cdot (e_{ij}^{\bullet,s} - f_{ij}^{\bullet,s})$

Example

We will do 3 iterations of LUCE and start with zero flow on the network. The first destination we will consider at iteration 1 is destination 3.
As example we will calculate the cost and its derivative of link 12, the cost and derivative of all links can be found in Figure 4.6.1. $c_{12} = \max\{c_{12}(f_{12}), \epsilon\} = \max\{5 + (f_{12})^2, \epsilon\} = \max\{5 + 0^2, \epsilon\} = 5$ and $g_{12} = \max\{\partial c_{ij}(f_{ij}), \epsilon\} = \max\{2 \cdot f_{12}, \epsilon\} = \max\{2 \cdot 0, \epsilon\} = \epsilon$.

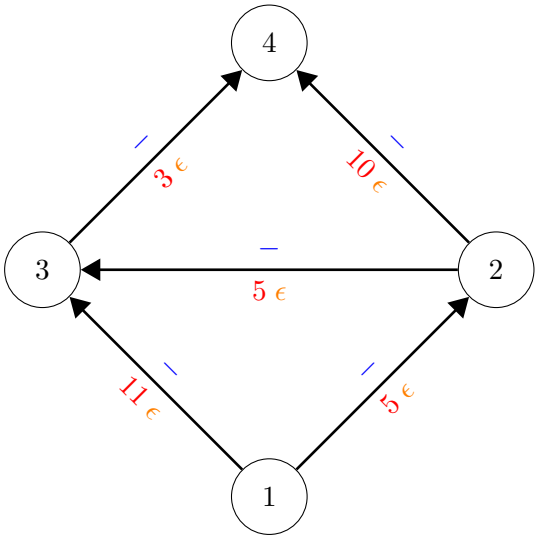


Figure 4.6.1: The cost and derivative of every link in a flowless network. The blue "-" means there is no flow. The red number below a link is the cost and the orange number is the derivative of the cost of that link.

All flows are zero, so we have $y_{ij}^3 = 0$ for all $ij \in A$.
There is no bush, so we have to initialize it. We are now looking for all links ij in the network with the property $C_i^3 \geq C_j^3 + c_{ij}$, where C_i^3 is the cost of a minimum path between i and 3. This is the case for links 12 and 23 ($C_1^3 = 10 = c_{12} + C_2^3$, $C_2^3 = 5 = c_{23} + C_3^3$). The bush of destination 3 consists of $B(^3) = \{12, 23\}$.
We can go on with calculating $\tilde{C}_i^3$, $\tilde{G}_i^3$ et cetera, but since there is only one path we know the demand from 1 to 3, $D_{1,3}$, will be assigned entirely to this path.
We are calculating the first iteration ($n = 1$), so $\lambda = 1$ and the flow is updated with $f_{ij} = f_{ij} + 1 \cdot (e_{ij}^{\bullet,3} - f_{ij}^{\bullet,3}) = 0 + 1 \cdot (e_{ij}^3 - 0) = e_{ij}^3$ and $f_{ij}^{\bullet,3} = f_{ij}^{\bullet,3} + 1 \cdot (e_{ij}^{\bullet,3} - f_{ij}^{\bullet,3}) = 0 + 1 \cdot (e_{ij}^{\bullet,3} - 0) = e_{ij}^{\bullet,3}$.
We get $f_{ij} = f_{ij}^{\bullet,3} = 2$ for $ij = 12$ and $ij = 23$ and $f_{ij} = f_{ij}^{\bullet,3} = 0$ for all other $ij \in A$. This flow can be found in Figure 4.6.2.

Now the demands of destination 3 are assigned we will look at destination 4, so $s = 4$.
The cost and derivatives can be found in Figure 4.6.2.
We have assigned flow to the network with destination 3, but the flow with destination 4 is zero, so $y_{ij}^4 = 0 \; \forall ij \in A$.
Searching for $C_i^4 \geq C_j^4 + c_{ij}$ gives links 13, 34 and 24. Again, there is only one path possible per OD-pair. The only path from 2 to 4 is $2-4$ and from 1 to 4 is $1-3-4$. Like destination 3

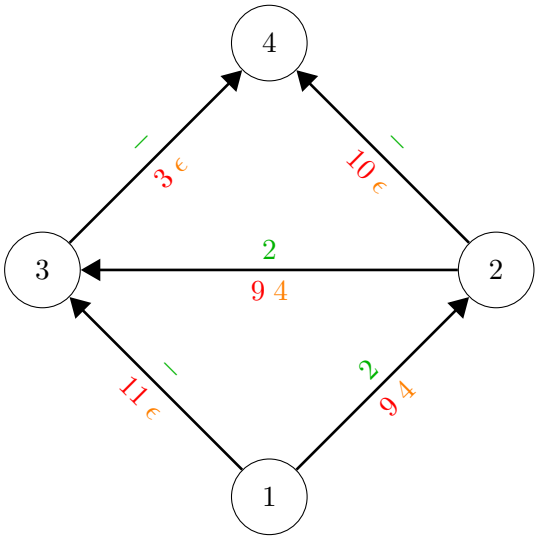


Figure 4.6.2: Assignment during iteration 1, when only destination 3 is considered. The green number above a link is the flow with destination 3 on that link. The red number below a link is the cost and the orange number is the derivative of the cost of that link.

there is only one path at iteration 1, so we get $f_{13}^{\bullet,4} = e_{13}^{\bullet,4} = 9$, $f_{34}^{\bullet,4} = e_{34}^{\bullet,4} = 9$, $f_{24}^{\bullet,4} = e_{24}^{\bullet,4} = 2$ and $f_{ij}^{\bullet,4} = e_{ij}^{\bullet,4} = 0$ for all other $ij \in A$. We have:

$f_{12}^{\bullet,3} = 2$ $f_{13}^{\bullet,3} = 0$ $f_{23}^{\bullet,3} = 2$ $f_{24}^{\bullet,3} = 0$ $f_{34}^{\bullet,3} = 0$
$f_{12}^{\bullet,4} = 0$ $f_{13}^{\bullet,4} = 9$ $f_{23}^{\bullet,4} = 0$ $f_{24}^{\bullet,4} = 2$ $f_{34}^{\bullet,4} = 9$

$f_{12} = 2$ $f_{13} = 9$ $f_{23} = 2$ $f_{24} = 2$ $f_{34} = 9$

$f_1 = 11$ $f_2 = 4$ $f_3 = 9$ $f_4 = 0$

These flows, together with the cost and cost derivative, can be found in Figure 4.6.3.

We want to calculate iteration 2 and begin this iteration with destination 3, meaning we have $n = 2$ and $s = 3$. The first step is calculating the cost c_{ij} and cost derivatives g_{ij}, which are given in Figure 4.6.3.
There is flow with destination 3 at the nodes 1 and 2, so we get $y_{12}^3 = f_{12}^{\bullet,3}/f_1^3 = 2/2 = 1$, $y_{23}^3 = 2/2 = 1$ and $y_{13}^3 = 0$, $y_{24}^3 = 0$ and $y_{34}^3 = 0$.
The bush of destination 3 consists of links 12 and 23, since these two links have flow. $C_1^3 \not\geq c_{13} + C_3^3$ meaning we do not add link 13 to the bush. The bush doesn't change and still has only one path. The flow can't change, thus we keep the same f_{ij}^3 and f_{ij}.

We go on with iteration 2 by equilibrating the bush of destination 4, thus we set $s = 4$.
We will now calculate y_{ij}^4:
$y_{12}^4 = 0$, $y_{23}^4 = 0$, $y_{13}^4 = f_{13}^4/f_1 = 9/9 = 1$, $y_{24}^4 = 2/2 = 1$, $y_{34}^4 = 2/2 = 1$
There is flow on link 13, 24 and 34, thus these links are part of the bush of destination 4. We have to add link 12 since it is part of the minimum cost path $1-2-3$ and thus $C_1^4 \geq C_2^4 + c_{12}$.
$B(^4) = \{13, 24, 34, 12\}$

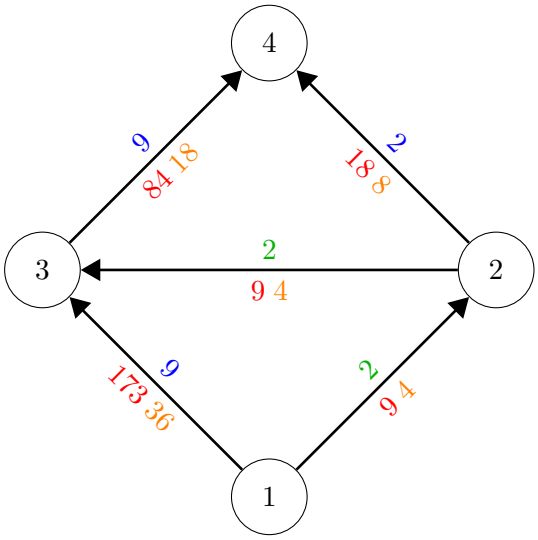


Figure 4.6.3: Assignment after 1 iteration of LUCE. The green number above a links is the amount of flow with destination 3 and the blue number the amount flow with origin 4. The red number below a link is the cost and the orange number is the derivative of the cost of that link.

$\tilde{C}_3^4 = 1 \cdot (84 + 0) = 84 \quad \tilde{C}_2^4 = 1 \cdot (18 + 0) = 18 \quad \tilde{C}_1^4 = 1 \cdot (173 + 84) = 257$
$\tilde{G}_3^4 = 1 \cdot (18 + 0) = 18 \quad \tilde{G}_2^4 = 1 \cdot (8 + 0) = 8 \quad \tilde{G}_1^4 = 1 \cdot (18 + 36) = 54$

To determine the alternative flow, we begin with setting it to zero: $e^4 = 0$
We fill in the demands $e_1^4 = 9$ and $e_2^4 = 2$.

We will look at each node in the bush in topological order. We begin with $i = 1$. The bush consist of $\{12, 24, 13, 34\}$, meaning the forward star bush of 1 is $J = \text{FSB}(1, 4) = \{2, 3\}$.
$V_1^4 = (e_1^{\bullet,4} + \sum_{j \in J}[(c_{1j} + \tilde{C}_j^4)/(g_{1j} + \tilde{G}_j^4)] / \sum_{j \in J}[1/(g_{1j} + \tilde{G}_j^4)] = 68.82$
$e_{12}^{\bullet,4} = V_1^4/(g_{12} + \tilde{G}_2^4) - (c_{12} + \tilde{C}_2^4)/(g_{12} + \tilde{G}_2^4) + e_1^4 \cdot y_{12}^4 = 3.48$
$e_2 = e_2 + e_{12}^{\bullet,4} = 2 + 3.48 = 5.48$
$e_{13}^{\bullet,4} = V_1^4/(g_{13} + \tilde{G}_3^4) - (c_{13} + \tilde{C}_3^4)/(g_{13} + \tilde{G}_3^4) + e_1^4 \cdot y_{13}^4 = 5.52$
$e_3^4 = e_3 + e_{13}^{\bullet,4} = 0 + 5.52 = 5.52$

We go on with $i = 2$ and $J = \text{FSB}(2, 4) = \{4\}$
$V_2^4 = (e_2^4 + \sum_{j \in J}[(c_{2j} + \tilde{C}_j^4)/(g_{2j} + \tilde{G}_j^4)] / \sum_{j \in J}[1/(g_{2j} + \tilde{G}_j^4)] = 38$
$e_{24}^{\bullet,4} = V_2^4/(g_{24} + \tilde{G}_4^4) - (c_{24} + \tilde{C}_4^4)/(g_{24} + \tilde{G}_4^4) + e_2^4 \cdot y_{24}^4 = 5.48$

The next node we consider is $i = 3$ and $J = \text{FSB}(3, 4) = \{4\}$
There is a demand of 2 travellers at node 3, all other flow arriving at node 3 must go to destination 4. We get $e_{34}^{\bullet,4} = 5.52$.

We now have determined the alternative flow. To determine λ, we have to calculate

$$\frac{\partial O(1)}{\partial \lambda} = \sum_{ij \in A} c_{ij}(f_{ij} + e_{ij}^s - f_{ij}^s) \cdot (e_{ij}^s - f_{ij}^s)$$

$\frac{\partial O(1)}{\partial \lambda} = -104.8 < 0$, thus we get $\lambda = 1$.

Filling in $\lambda = 1$ gives us $f_{ij} = f_{ij} + e_{ij}^{\bullet,4} - f_{ij}^{\bullet,4}$. This means that the new assignment of flow with destination 4 is exactly the alternative assignment, thus iteration 2 has Figure 4.6.4 as assignment.

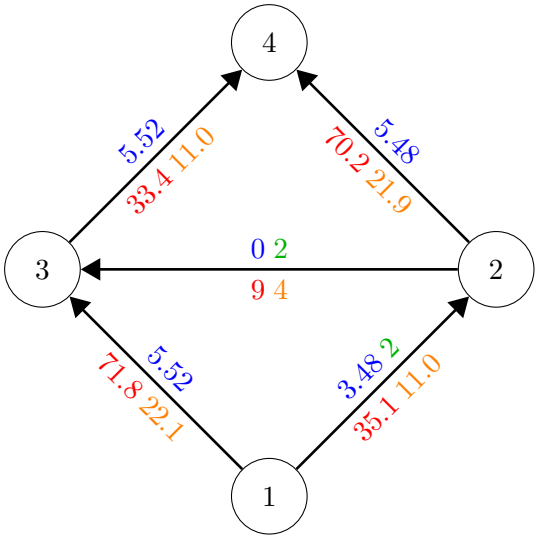


Figure 4.6.4: Assignment after 2 iterations of LUCE. The green number above a links is the amount of flow with destination 3 and the blue number the amount flow with origin 4. The red number below a link is the cost and the orange number is the derivative of the cost of that link.

We will look at iteration 3 and destination 3 ($n = 3$, $s = 3$). c_{ij} and g_{ij} can be found in Figure 4.6.4.
There is flow with destination 3 at nodes 1 and 2, giving $y_{12}^3 = 1$, $y_{23}^3 = 1$ and $y_{13}^3 = y_{24}^3 = y_{34}^3 = 0$
The bush is $B(^3) = \{12, 23\}$. We don't have $C_1^3 \geq C_3^3 + c_{13}$, thus we don't add link 13. The bush isn't modified and there is still only one path in the bush, so the flow won't change. We will consider the next destination.

We are still busy with iteration 3, but now with destination 4 ($n = 3$, $s = 4$)
c_{ij} and g_{ij} are still the same as in $n = 3$ and $s = 4$, since the network didn't change.
$y_{12}^4 = f_{12}^{\bullet,4}/f_1^4 = 0.39$, $y_{13}^4 = 0.61$, $y_{23}^4 = 0$, $y_{24}^4 = 1$ and $y_{34}^4 = 1$
The modified bush consists of $B(^4) = \{12, 13, 24, 34, 23\}$ since 23 is in the minimum cost path and thus $C_1^3 \geq C_3^3 + c_{13}$ and 12, 13, 24 and 34 were already in the bush.

$\tilde{C}_2^4 = \sum_{j \in \text{FSB}(2,4)} y_{2j}^4 \cdot (c_{2j} + \tilde{C}_j^4) = 1 \cdot (70.2 + 0) + 0 \cdot (9 + 33.4) = 70.2$ and $\tilde{G}_2^4 = \sum_{j \in \text{FSB}(2,4)} (y_{2j}^4)^2 \cdot (g_{2j} + \tilde{G}_j^4) = 1^2 \cdot (21.9 + 0) + 0^2 \cdot (4 + 11.0) = 21.9$. $\tilde{C}_i^4$ and $\tilde{G}_i^4$ $\forall i \in I$ are stated here:

$\tilde{C}_4^4 = 0$ $\tilde{C}_3^4 = 33.4$ $\tilde{C}_2^4 = 70.2$ $\tilde{C}_1^4 = 105.3$
$\tilde{G}_4^4 = 0$ $\tilde{G}_3^4 = 11.0$ $\tilde{G}_2^4 = 21.9$ $\tilde{G}_1^4 = 33.0$

We are now going to determine the alternative flow e.

We set the alternative flow to zero and then assign the demands to the origin node: $e_1^4 = 9$ and $e_2^4 = 2$

Determine the forward star bush and alternative flow for every node in the bush, except the destination node, in topological order.
The first node is $i = 1$ with forward star bush J =FSB$(1, 4) = \{2, 3\}$
$V_1^4 = 105.25$
$e_{12}^{\bullet,4} = 3.48$ $e_{13}^{\bullet,4} = 5.52$ $e_{23}^{\bullet,4} = 0.75$ $e_{24}^{\bullet,4} = 4.73$ $e_{34}^{\bullet,4} = 6.27$
$e_1^4 = 9$ $e_2^4 = 5.48$ $e_3^4 = 6.27$
as can be seen in Figure 4.6.5.

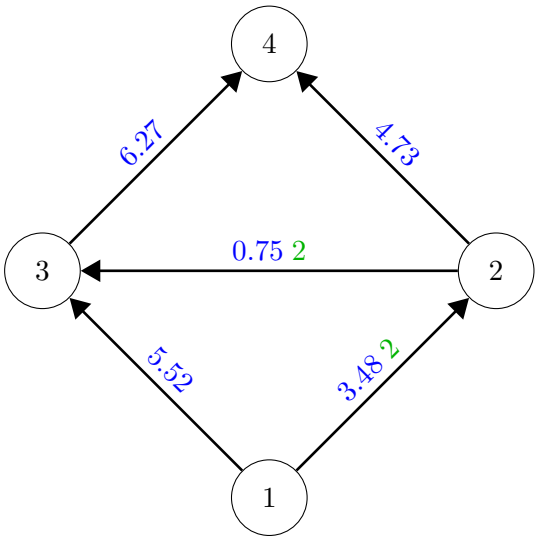


Figure 4.6.5: Alternative flow at iteration 3 of LUCE. The green number above a link is the flow with destination 3 and the blue with destination 4 of that link.

$\partial O(1) \,/\, \partial\lambda = 35.5 > 0$ and $\partial O(0)/\partial\lambda = -20.8 < 0$.
We get $\lambda = \dfrac{1}{1 - (\partial O(1) \,/\, \partial\lambda) \;/\; (\partial O(0) \,/\, \partial\lambda)} = 0.37$

Filling in the λ gives Figure 4.6.6.

Properties

Memory usage	+−
Proportionality	−
Rate of convergence	+

4.7 Algorithm B (Alg. B)

Algorithm B is a bush-based algorithm, which uses the minimum cost tree and maximum cost tree to determine segments. Within these segments the flow is changed in order to equilibrate the cost.

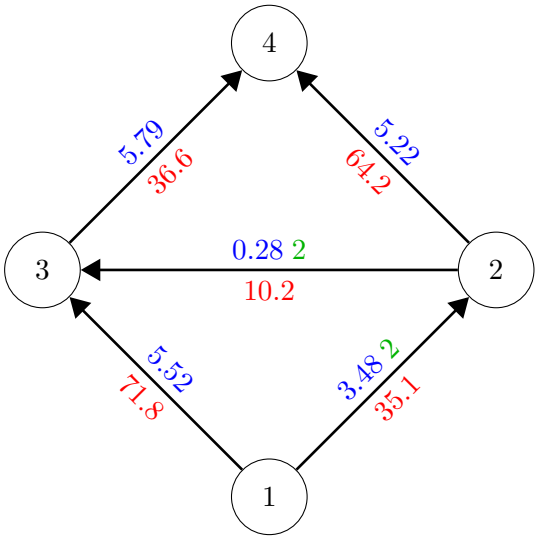


Figure 4.6.6: Assignment after 3 iterations of LUCE. The green number above a link is the flow with destination 3 *and the blue with destination* 4 *of that link. The red number below a link is the cost of that link.*

Description

This algorithm starts with an initial feasible bush. This can be determined by doing an AON-assignment on a flowless network and initialize the bush or by using an initial assignment where the bush is known. The algorithm continues with selecting an origin. Next one determines the minimum cost paths between the origin and every destination, also known as the minimum path tree. Simultaneously the maximum path tree is determined, which are all maximum cost paths within the bush from the origin to all destinations. One searches for "pairs of alternative segments" (PASs) where one part of a PAS is an element of the minimum path tree and the other part is an element of the maximum path tree. If the bush is not optimal when the PASs are equilibrated then the bush is modified by adding the cheapest path(s), otherwise another origin is selected to equilibrate its bush.

Example

We will give an example of this algorithm on the network of Figure 4.0.1.
We begin with making an AON-assignment and determine the bush and modified bush. The realization of the AON-assignment can be found at the beginning of this section. In Figure 4.7.1 the thick black arrows mark the maximum path tree and the blue lines the minimum path tree of origin 1. The bush consists of all the paths with flow and the modified bush consists of the bush plus the minimum cost tree.

We can see a PAS, PAS1, in the lower triangle $(1-2-3, 1-3)$: with segment $1-2-3$ as element of the maximum path tree and segment $1-3$ as element of the minimum path tree. This is the only PAS with one segment as part of the minimum and one segment as part of the maximum path tree. We are going to shift x flow with origin 1 within this PAS, so we

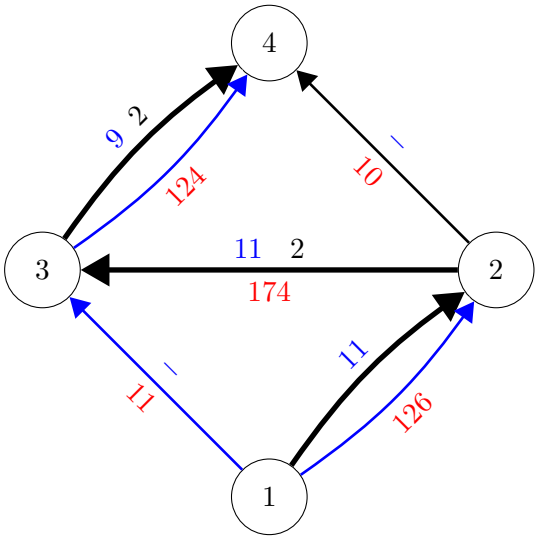


Figure 4.7.1: Origin-based assignment of Figure 4.0.3. The red number below a link is the cost of that link. The blue number above a link is the flow with origin 1 *and the black number is the flow with origin* 2*. The thick black arrow marks the maximum cost tree and the blue arcs the minimum cost tree of origin* 1*.*

can shift at most 11 travellers. We have

$$\min_{0\leq x\leq 11} \int_0^{11-x} c_{12}(w)\,\mathrm{d}w + \int_0^{0+x} c_{13}(w)\,\mathrm{d}w + \int_0^{2+11-x} c_{23}(w)\,\mathrm{d}w$$

which has $x = 6.02$ as result. Updating the network gives Figure 4.7.2.

We had only one PAS and this one is equilibrated, but when we look at the whole network origin 1 is not equilibrated. We can see that path $1-2-4$ costs far less than path $1-3-4$. The next step is searching the minimum and maximum path tree. These are already shown in Figure 4.7.2 by thick black arcs and blue arcs. We can see the lower triangle is equilibrated but the upper triangle $(2-3-4, 2-4)$ is not. This upper triangle, PAS2, has segment $2-3-4$ as part of the maximum path tree and the segment $2-4$ as part of the minimum path tree. We are now going to equilibrate PAS2 by changing x flow with origin 1 within this PAS. The maximum flow shift is 4.98, since this is the amount of flow on the expensive segment where we extract flow from. Minimizing

$$\min_{0\leq x\leq 4.98} \int_0^{2+4.98-x} c_{23}(w)\,\mathrm{d}w + \int_0^{0+x} c_{24}(w)\,\mathrm{d}w + \int_0^{2+9-x} c_{34}(w)\,\mathrm{d}w$$

gives $x = 4.66$. Shifting this x flow results in Figure 4.7.3.

By equilibrating PAS2 we changed a segment of PAS1, annulling the equilibrium of PAS1. We can see this in the maximum path tree and minimum path tree of Figure 4.7.3, where a link in the lower triangle is used which is not part of the minimum path tree. We are equilibrating PAS1 again by changing the flow of origin 1. The result can be found in Figure 4.7.4.

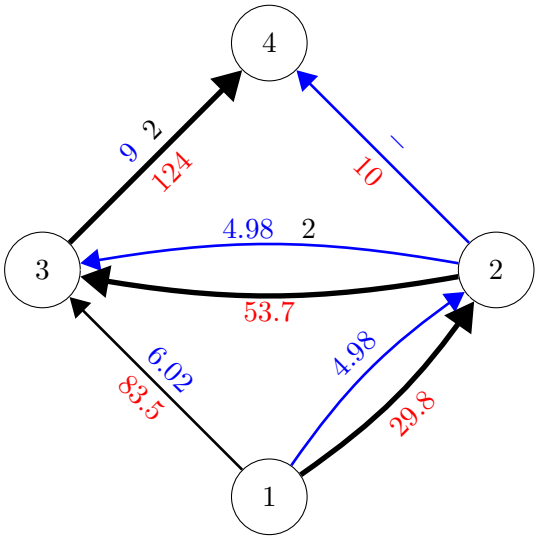


Figure 4.7.2: Assignment after 1 iteration of Alg. B. The red number below a link is the cost of that link. The blue number above a link is the flow with origin 1 *and the black number is the flow with origin* 2*. The thick black arrow marks the maximum cost tree and the blue arcs the minimum cost tree of origin* 1*.*

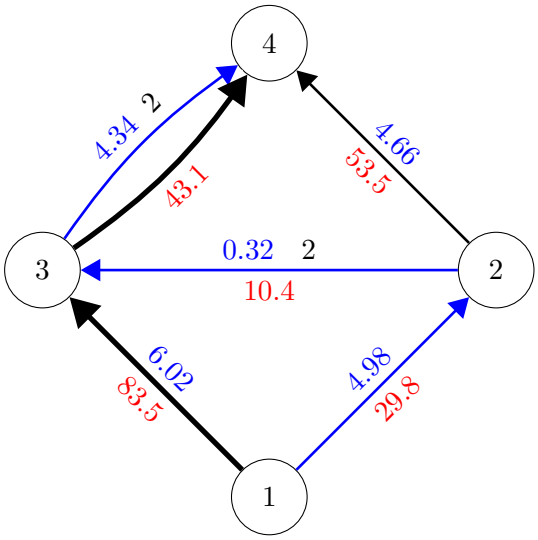


Figure 4.7.3: Assignment after 2 iterations of Alg. B. The red number below a link is the cost of that link. The blue number above a link is the flow with origin 1 *and the black number is the flow with origin* 2*. The thick black arrow marks the maximum cost tree and the blue arcs the minimum cost tree of origin* 1*.*

This algorithm will go on with changing flow with origin 1 in alternating PAS1 and PAS2. We began with origin 1 and will go to origin 2 when the bush of origin 1 is in equilibrium. In our example the flow with origin 2 will never change, since when the bush of origin 1 is in equilibrium the entire network will be in equilibrium.

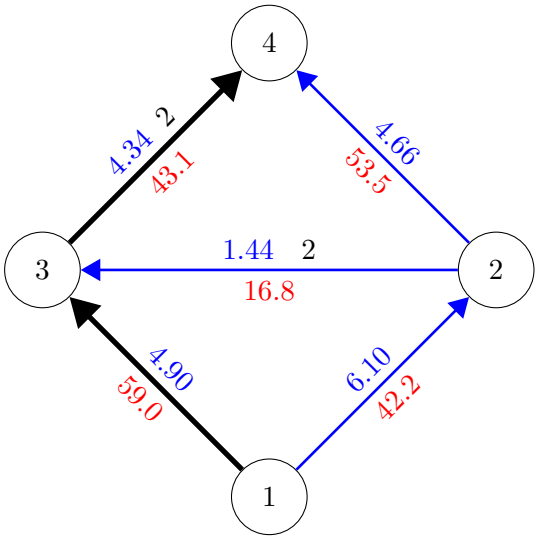


Figure 4.7.4: Assignment after 3 iterations of Alg. B. The red number below a link is the cost of that link. The blue number above a link is the flow with origin 1 *and the black number is the flow with origin* 2*. The thick black arrow marks the maximum cost tree and the blue arcs the minimum cost tree of origin* 1*.*

Properties

Memory usage	+−
Proportionality	−
Rate of convergence	+

4.8 Traffic assignment by paired alternative segments (TAPAS)

As the name of this algorithm suggests, TAPAS uses PASs to converge to the optimal assignment. The algorithm is a bit similar to Alg. B, but "the main differences from Dials algorithm (Algorithm B red.) are: the procedure to identify PASs which is based on Bar-Gera (2006) ([14] red.); there are no restrictions to a specific bush; PASs are stored from iteration to iteration; and all relevant origins for each PAS are considered" [6].

Description

We will first give a sketch about how the algorithm works, next the description as in the paper [6] is cited.

The main idea of this algorithm is equilibrating cost in a PAS and redistribute the flow in this PAS between the relevant origins. The algorithm uses PASs, where a PAS consists of two alternative paths where one path has low costs and the other path has higher costs. A PAS is constructed by finding two segments with the same begin node and the same end node where one segment is part of a minimum cost tree and the other segment is used by the origin you are currently investigating and not part of a minimum cost tree. When equilibrating cost in a PAS one changes the flow of multiple relevant origins at the same time. We try to find a PAS with as least links as possible, since we try to change flow of as many origins as possible.

It would take relatively much time to determine every iteration, for every PAS, which origins it uses; therefore the origins are marked as relevant for a PAS.

The algorithm starts with an initial solution. One has to investigate all origins one by one and search for links that are used by the origin one is currently investigating but are not part of a minimum cost path. A PAS is needed for these links. Either there was already a PAS using this link, then the origin is marked as relevant for this PAS, or there was no such PAS yet and a new PAS is constructed. Next the cost is equilibrated within a PAS and the flow is redistributed between the relevant origins by the proportionality conditions.

The pseudo code for this algorithm is [6]:

```
Find initial solution using all or nothing assignment
Repeat iteratively
        For every origin
            Remove all cyclic flows
            Find tree of minimum cost routes
            For every link used by the origin which is not part of the tree
                If there is an existing effective PAS
                  Mark the origin as relevant for this PAS
                Else
                  Construct a new PAS
                  Mark the origin as relevant for this PAS
            Choose a random subset of active PASs
                Shift flow within each chosen PAS
        For every active PAS
            Check if it should be eliminated
            Perform flow shift to equilibrate costs
            Redistribute flows between origins by the proportionality condition
Final proportionality iterations:
        For every active PAS
            Redistribute flows between origins by the proportionality condition
```

Summarizing, we have that the iterations consists of investigating an origin for PASs, shift flow in a random subset of PASs and investigate the next origin until every origin is looked at. The iteration ends with shifting flow for every active PAS, which can be done multiple times. We will do these iterations until the stopping criterion is met. When this criterion is met, we will end the algorithm with performing some proportionality iterations for every active PAS.

Example

We will do two iterations of TAPAS to make clear how it works. We assume there is no special junction at node 3 of our example network.

TAPAS starts with an initial assignment. Ours is the AON-assignment on a zero flow network, which is drawn in Figure 4.7.1. We will now determine all possible PASs. There are two

alternative paths from origin 1 to node 3, namely $1-2-3$ and $1-3$. These paths are called segments and together they form a pair of alternative segments, or shortly a PAS. There are also two paths from 2 to 4: $2-3-4$ and $2-4$. There are three paths from 1 to 4, so one might think that these three paths result in 3 PASs, but this is not true. One condition of a PAS is that the two segments have no common link. When we shift flow from $1-2-3-4$ to $1-2-4$, the flow on link 12 does not change, since this link is part of both segments. A possible PAS for these paths consists of the segments $2-3-4$ and $2-4$, which was already found. The three paths from 1 to 4 results in one new PAS with the segments $1-2-4$ and $1-3-4$. The example network has 3 possible PASs, which are coloured in Figure 4.8.1. PAS1 is blue, PAS2 is red and PAS3 is green.

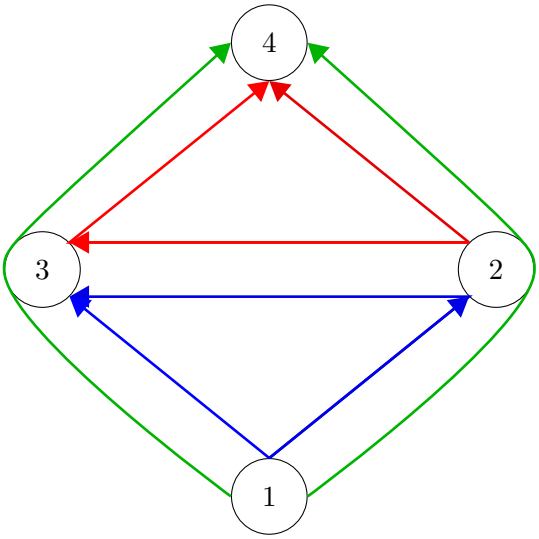


Figure 4.8.1: The three possible PASs of our example network. PAS1 is blue, PAS2 red and PAS3 green.

We consider origin 1.
There are no cyclic flows, since all links are directed in such a way that no cyclic flow can be made.
Links 12, 23 and 34 are used by origin 1. The minimum cost paths are $1-3$ for OD-pair 13 and $1-3-4$ for OD-pair 14. The tree of minimum cost routes is $\{13, 34\}$. We can see that the used paths and minimum cost paths are different at links 13, 12 and 23. This is exactly PAS1 (blue PAS). The set of active PASs consists of PAS1, where the relevant origin of PAS1 is origin 1. We shift flow for a random subset of PASs. To make this example not too large, we assume the random subset is empty for all iterations. Notice that although the random subset is empty, we will perform a flow shift for every active PAS at the end of every iteration.

We consider origin 2 now. There is no cyclic flow.
Path $2-3-4$ is used where $2-4$ is the minimum cost path. These two paths are both a segment in PAS2. We add PAS2 to the set of active PASs and mark origin 2 as relevant origin for PAS2. Suppose our subset of active PASs is empty.

For every active PAS we have to check if it should be eliminated, shift flow and redistribute the flow between the relevant origins. The PASs are just found, so we do not eliminate them. We begin with shifting flow between the two segments within PAS1 to equilibrate the cost. This means

$$\min_{0\leq x\leq 11} \int_0^{11-x} c_{12}(w)\,\mathrm{d}w + \int_0^{0+x} c_{13}(w)\,\mathrm{d}w + \int_0^{2+11-x} c_{23}(w)\,\mathrm{d}w$$

The answer to this minimization problem is $x = 6.02$. We don't have to do any redistribution since there is only one relevant origin. Our current answer is the same as in iteration 1 of Alg. B, which can be found in Figure 4.7.2. Notice that in this figure the thick black arrows mark the maximum path tree of origin 1 and not the minimum cost tree of origin 2.

We want to equilibrate the cost of PAS2 now. The only relevant origin is origin 2, which has a flow of 2 travellers at the higher cost segment. This means we can only change 2 this iteration. Minimizing the objective function gives as result to move all 2 to the lower cost segment, as can be seen in Figure 4.8.2.

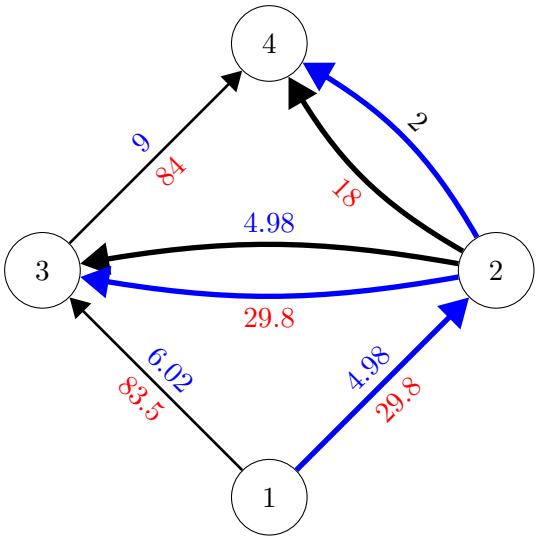


Figure 4.8.2: Assignment after 1 iteration of TAPAS. The blue number above a link is the flow with origin 1*, the black number the flow with origin* 2 *and the red number below a link is the cost. The thick blue arrows are the minimum cost paths of origin* 1 *and the thick black arrows the minimum cost paths of origin* 2*.*

We can equilibrate the PASs a few more times, but to keep this example small we will not do this. We have investigated all origins and done a flow shift for every active PAS, so we go on with iteration 2. We start iteration 2 with establishing that there is no cyclic flow in the bush of origin 1. The tree of minimum costs of origin 1 is $\{12, 23, 24\}$. The links used by origin 1 which are not part of the minimum cost tree are 13 and 34. 13 is part of PAS1 and 34 of PAS2. Origin 1 is relevant for PAS1 and PAS2.
Since we assumed the subset is empty, we continue with origin 2. The minimum cost tree is $\{24\}$ where the used links are $\{23, 34\}$. These links are part of PAS2, so origin 2 must be marked as relevant for PAS2, which was already done.

We have to check if a PAS should be eliminated and if not, equilibrate the cost and redistribute the flow between the relevant origins. Both PASs are used the previous iteration, thus they should not be eliminated. We begin the flow shift for PAS2. We now have both origins marked as relevant, so we can change flow of both origin 1 as origin 2. The maximum flow we can shift is 4.98. Shifting flow to equilibrate the cost and redistributing the flow between the relevant origins gives Figure 4.8.3.

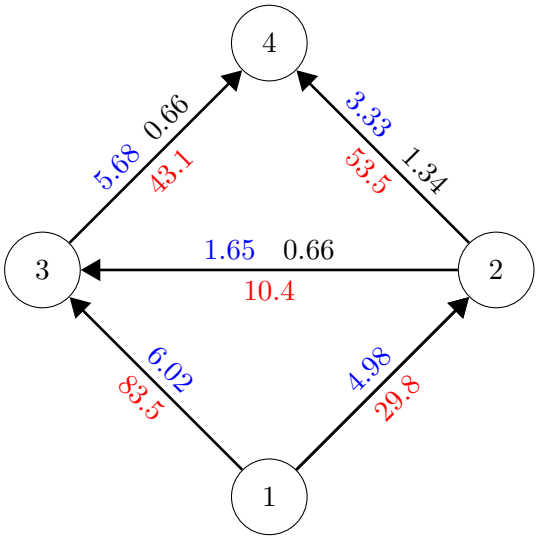


Figure 4.8.3: Assignment during iteration 2, when the cost of PAS2 is just minimized. The blue number is the flow with origin 1 *and the black number above a link is the flow with origin* 2*. The red number below a link is the cost.*

The last part of iteration 2 is equilibrating the cost in PAS1. We shift flow from the more expensive segment, $1-3$, to the cheaper segment $1-2-3$ in order to equalize the cost of both paths. The result can be found in Figure 4.8.4.

After we have done N iterations or the relative gap is small enough, we stop with equilibrating PASs and do a predetermined amount of proportionality iterations. A proportionality iteration exists of redistributing the flow between the relevant origins for every active PAS, where the flow redistribution is according to the proportionality condition.

Properties

Memory usage	$-$
Proportionality	$+$
Rate of convergence	$+$

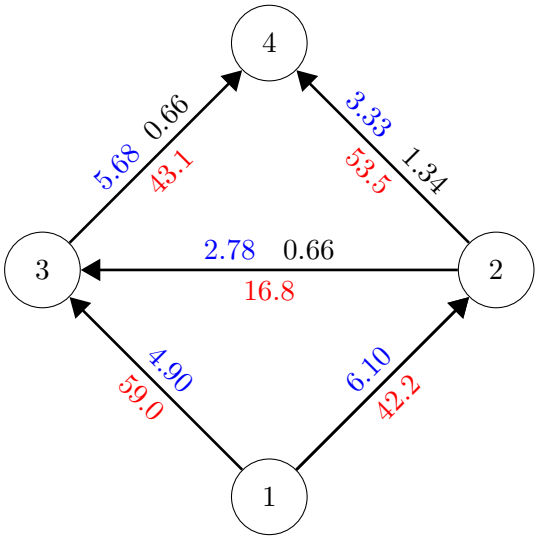


Figure 4.8.4: Assignment after 2 iterations of TAPAS. The blue number is the flow with origin 1 *and the black number above a link is the flow with origin* 2*. The red number below a link is the cost.*

5 Modified TAPAS

In this chapter we will first unfold why we have chosen TAPAS to look closer at. Next our modifications to TAPAS are explained and then the additions to TAPAS to include junction modelling.

5.1 Why TAPAS

The ideal algorithm would only have "+" in the property table. Unfortunately, there is no algorithm known that uses very little memory and converges fast while maintaining proportionality. We have to make a compromise between the desirable properties. We want a faster converging algorithm than the current algorithms in OmniTRANS: IA, MSA and FW. The algorithms PG, LUCE, Alg. B and TAPAS have all converge rates in the same order of magnitude. The cost function for a link is often the BPR-function, but we want the possibility to choose any increasing function, which is not possible in LUCE. One of the discussed properties in Chapter 4 is proportionality, which is in our opinion an important property. PG and Alg. B are not proportional, which leaves us with TAPAS as possible choice. The disadvantage of TAPAS is the more than average memory usage, but the amount of memory in PC's and laptops is increasing fast, making the memory usage less and less a restriction. Junction modelling was not proposed for TAPAS before, but this is described in Section 5.3.

5.2 Small modifications to TAPAS

We can have PASs where the cost difference between the 2 segments are almost zero. One can consider to stop with determining the flow shift to equalize the cost and go on with shifting flow in another PAS in order to spend the time "more useful". These and other considerations can be found in Section 5.2.1. The determination of the approximated RGAP can be found in Section 5.2.2 and in the last part of this section we say something about the random subset and reproducibility.

5.2.1 Shift flow

It is not always possible to shift much flow in a PAS. This can be caused by a badly chosen PAS or by the fact that the flow is too spreaded to have much flow on a link. The solutions proposed in the paper [6], which will be explained next, require flow-effective PASs, cost-effective PASs and making branch shifts if there is no cost-effective PAS.

A PAS consists of 2 segments. Suppose segment 1 is the one with a higher cost and the other segment is segment 2, where segment 1 ends with link $\hat{\imath}j$ and segment 2 with $\tilde{\imath}j$. This can be found in Figure 5.2.1

A PAS is called cost-effective for origin r if $c_{\{segment1\}} + c_{\{segment2\}} \geq \eta \cdot rc_{r,ij}$, link $\tilde{\imath}j \in$ minimum cost tree and $\hat{\imath}j \notin$ minimum cost tree, where η is a constant between 0 and 1 and the reduced cost is $rc_{r,ij} = C_r^{\hat{\imath}} - C_r^j + c_{\hat{\imath}j}$.
A disadvantage is that we have to determine the cost of both segments for many PASs. The other part, determining if $\hat{\imath}j$ and $\tilde{\imath}j$ are in the minimum cost tree, is less difficult since we look for PASs after the tree of minimum cost routes is determined. Which links are part of a minimum cost route is already known. If no existing PAS is cost-effective one can construct

a new PAS which is automatically cost-efficient, since the lower cost segment is always part of the minimum cost tree.

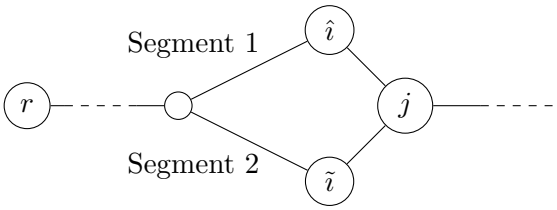
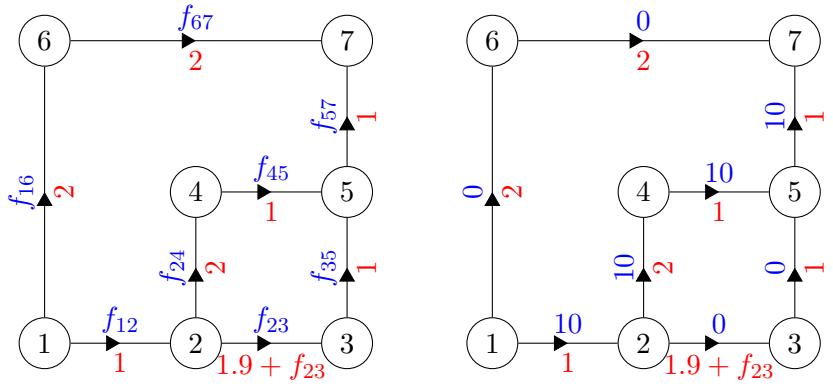


Figure 5.2.1: The notations used for the explanation of effective PASs. Circles are nodes and the upper segment is segment 1 *and the lower segment is segment* 2*.*

A PAS is effective if it is not only cost-effective but also flow-effective, where a PAS is flow-effective if $\min_{ij \in \text{segment } 1} f_{ij}^{r,\bullet} \leq \gamma \cdot f_{\hat{i}j}^{r,\bullet}$, with $0 \geq \gamma \geq 1$. If a PAS is not flow-effective, one has to search for a flow-effective PAS. Due to flow spreading there may not exist such a PAS. In this case a branch shift is recommended in the paper. A branch is a set of segments from an origin ending with a single link. A branch shift means that instead of shifting flow from one path to the other path we shift flow from multiple paths (a branch) to the other path. The paper [6] does not describe how branches can be found. The three solutions, cost-effective PAS, flow-effective PAS and branch shifts, help to identity PASs which are almost equilibrated or have almost no flow. One can say that requiring effective PASs and allow branch shifts help to ignore "not interesting" PASs. These requirements are the basis of the proof of convergence of TAPAS. We will give a little example to show why we would require flow-effective PASs.
Suppose we have the network of Figure 5.2.2a and we have PAS1 $\{1-2-3-5-7, 1-6-7\}$ and PAS2 $\{2-3-5, 2-4-5\}$. Equilibrating PAS2 will lead to a shift of 0.1 and then PAS1 will shift this 0.1 to path $1-6-7$. This will continue until $f_{\{2-4-5\}} = 0$, which takes 100 iterations. When we have $c_{23} = 2-\epsilon$ the number of iterations needed until the UE is found can even be much larger.



(a) Network. (b) Initial assignment.

Figure 5.2.2: Example to show that requiring effective PASs may be beneficial. Red formula's/numbers are the cost and the blue formula's/numbers the flow of that link.

PASs are eliminated when they are not used for three iterations. A PAS will also stay in memory when only a little bit of flow is shifted every iteration. So PASs can be saved while they are not "interesting". We will consider next some options which lead to an earlier eliminition of PASs.

When we have a PAS where we want to shift flow from the expensive segment to the cheaper segment, we have to go through a few steps. First we determine the cost of both segments in order to know which segment is the more expensive one. Next we determine the maximum amount of flow we can shift. Then we determine how much flow we have to shift in order to equalize the cost and at last we shift them. We mark the PAS with the iteration number to make sure the PAS is not eliminated the next iteration. We call all these steps together the shift flow function.
A modification of TAPAS is that after each step we can decide to go on to the next step or to stop with the shift flow function for this PAS and go further with TAPAS. The first step is determining the cost of the segments. We can say that when there is a cost difference of at least ϵ_2 we go to the next step and otherwise we stop with the shift flow function for this PAS. Another possibility is to make a restriction on the maximum amount of flow we can shift. When we can shift maximal ϵ_3 flow we stop with the shift flow function. Next we determine the amount of flow we want to shift in order to equilibrate the cost. Also here we can say that when this is less than an epsilon, ϵ_4, we stop with this shift flow function and when it is more we do shift the flow.

We will now give a little example to make the previous subsection more clear.
Suppose we have the network of Figure 5.2.3, where the blue line marks segment 1, the expensive segment, and the green line segment 2. Assume $\epsilon_2 = \epsilon_3 = \epsilon_4 = 0.1$. First we calculate the cost of both segments. These are already given: 65 and 5 for respectively segment 1 and 2. The cost difference is $65 - 5 = 60$. This is greater than ϵ_2, thus we go on with determining the maximum possible shift. The blue segment, segment 1, is the expensive one. This segment has a flow of 8, so we can shift at most 8. We go on with determining the shift since $8 > \epsilon_3$. The segments have equal cost when we shift 3. Also this number is greater than its epsilon, $3 > \epsilon_4$, thus we perform the flow shift.

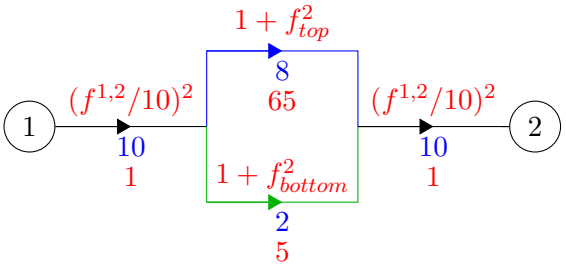


Figure 5.2.3: Example to make the shift flow function more clear. The red formula is the cost function where f_{top} is the flow on the upper (blue) segment and f_{bottom} of the lower (green) segment, the red number is the cost and the blue number the flow of that link. We have $D_{1,2} = 10$.

In the example we had ϵ_2, ϵ_3 and ϵ_4 equal, but this is not necessary. They can be a constant but also dependent on, for example, iteration number, RGAP or the amount of flow shifted during the previous iteration. In this way one can prevent that flow shifts are performed on PASs which are almost equilibrated while one should forget about these PASs. We suggest to make the epsilons dependent on the RGAP or maximum flow shifted during the last iteration(s).

When we have determined how much flow we can shift, we have done the biggest part of the shift flow function. When the flow to shift is less than ϵ_4 one can say it is a pity to have done these calculations and not using them. In this case we can "cheat". Instead of stopping with the shift flow function with this PAS we can do the flow shift, but not mark the PAS as used in this iteration. In this way the PAS is not stored longer than necessary and the assignment is closer to equilibrium.

The advantage of the epsilons is that it takes almost no extra computation time while deleting "not interesting" PASs iterations earlier.

5.2.2 Determination relative gap (RGAP)

We recall the definition of the relative gap (RGAP):

$$1 - \frac{\text{total minimum cost}}{\text{total cost}} = 1 - \frac{\sum_{r \in R} \sum_{s \in S} C_r^s D_{r,s}}{\sum_{ij \in A} f_{ij}\, c_{ij}}$$

where C_r^s is the cost of the minimum cost route from origin r to destination s, $D_{r,s}$ is the demand from r to s, f_{ij} is the flow on link ij and c_{ij} is the cost of this link.

At the end of an iteration we calculate the RGAP to determine whether we have to do another iteration or not. We stop the iterations when either we have reached the maximum number of iterations or RGAP $< \epsilon$. We need the cost of the shortest paths in order to calculate the RGAP. We can calculate this at the end of the iteration, as described in the paper, but we can also use an approximation which takes less time to compute. We calculate the cost of the shortest paths from an origin to all nodes when we determine the tree of minimal cost routes for that origin. After the tree of minimal cost routes is determined the flows and costs will change, causing a change in the cost of the minimum cost paths. One can see the cost used in the approximation as the minimum cost of the previous iteration. The minimum cost will in general increase during the iterations, producing an approximation which is smaller than the real minimum cost. This results in an approximated RGAP which is greater than the real RGAP, meaning that in general the precision will be a bit better than ϵ. To make sure we do not end the iterations before the RGAP is small enough we determine the real minimum cost when the approximated RGAP is smaller than ϵ.

We can summarize the above as:

if approximated RGAP $< \epsilon$
  determine minimum cost

```
  if RGAP < ϵ
    stop with iteration, go to proportionality iterations
  else
    do the next iteration
  end
else
  do the next iteration
end
```

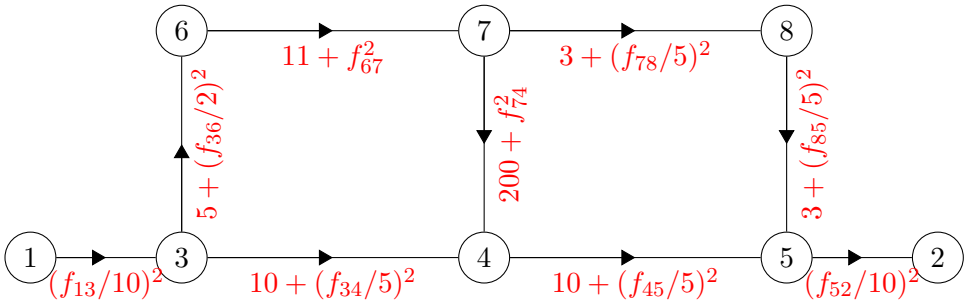
Due to the fact that TAPAS converges to a UE, one might think that the minimum cost is an increasing sequence, and thus the RGAP is a lower bound for the approximated RGAP, or that the total cost is a decreasing function. These will be shown to be untrue in the next two examples.
The network used to show a decreasing minimum cost can be found in Figure 5.2.4a. Suppose our initial assignment is as given in Figure 5.2.4b. The shortest path from the origin to the destination is marked by the thick blue line. The total cost is $10 \cdot 1 + 0 \cdot 10 + 10 \cdot 14 + 10 \cdot 1 + 10 \cdot 30 + 10 \cdot 300 + 0 \cdot 3 + 10 \cdot 111 + 0 \cdot 3 = 4570$. The minimum cost is $10 \cdot (1 + 10 + 14 + 1) = 260$. When searching for PASs we will find the segments $3 - 4$ and $3 - 6 - 7 - 4$, marked by the green line. When we shift flow in this PAS we get the flow as drawn in Figure 5.2.4c. Now we have a total cost of $10 \cdot 1 + 10 \cdot 14 + 10 \cdot 14 + 10 \cdot 1 + 0 \cdot 5 + 0 \cdot 200 + 0 \cdot 3 + 0 \cdot 11 + 0 \cdot 3 = 300$ and a minimum cost of $10 \cdot (1 + 5 + 11 + 3 + 3 + 1) = 240$. This gives an RGAP $1 - \frac{240}{300} = 0.20$. The approximated RGAP is $1 - \frac{260}{300} \approx 0.13$. In this case the minimum cost decreased, which resulted in a approximated RGAP which was smaller than the RGAP.

We have the following example to show that the total cost can increase. Our network can be found in Figure 5.2.5a. In Figure 5.2.5b we can see that the total cost is $10 \cdot 100 + 1 \cdot 123 = 1123$ and the minimum cost is $11 \cdot 100 = 1100$. There is only one PAS, which is marked in green. When we shift flow in this PAS we get Figure 5.2.5c. The total cost in this figure is $11 \cdot 121 = 1331$, which is also the minimum cost. The RGAP was $1 - \frac{1100}{1123} = 0.02$ and after the flow shift it is $1 - \frac{1131}{1131} = 0$. In this example the total cost has increased. Although the total cost increased the RGAP decreased, due to the increased minimum cost.

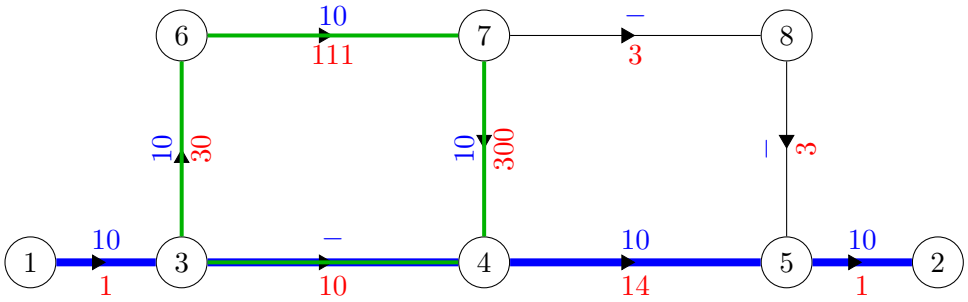
5.2.3 Random subset

Omnitrans International and users of OmniTRANS want a reproducible assignment, which means that when one uses the same input one always obtains the same output. The UE in terms of link flow is unique, but the algorithms are often stopped before equilibrium is reached. One part of TAPAS is taking a random subset of PASs and shift flow within these PASs. If we would take a random subset we will not get the same answer every time we use the same input. One way to solve this problem is using a seed. A seed always generates the same "random" string of numbers. In this case we have the profits of a random subset, but we also get the same answer for the same input.

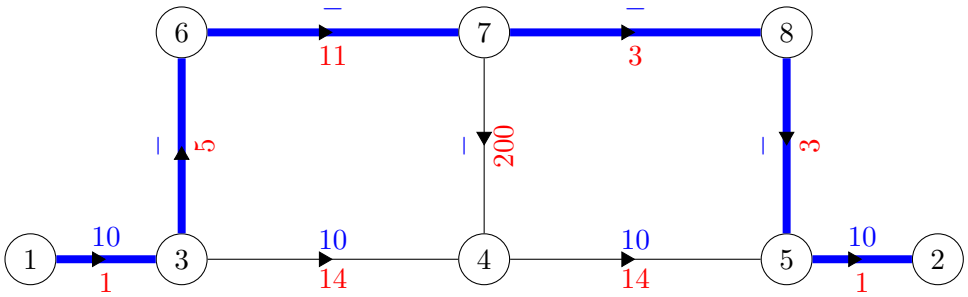
Bar-Gera takes the size of the subset equal to 20. This number is determined empirically. Since networks can vary much in the number of OD-pairs, nodes and links, it may be better to determine the size of the subset as a function of the number of OD-pairs, nodes, links or PASs. For the $n \times n$ grid network we used in Matlab, we used a subset size of n.



(a) Network.



(b) Initial flow, where the green lines mark a PAS.



(c) Flow after one iteration.

Figure 5.2.4: Example to show that the minimum cost can decrease. The red formula near a link is the cost function, the red number the cost and the blue number the flow on that link. The blue link marks the minimum path between 1 *and* 2*. We have* $D_{1,2} = 10$*.*

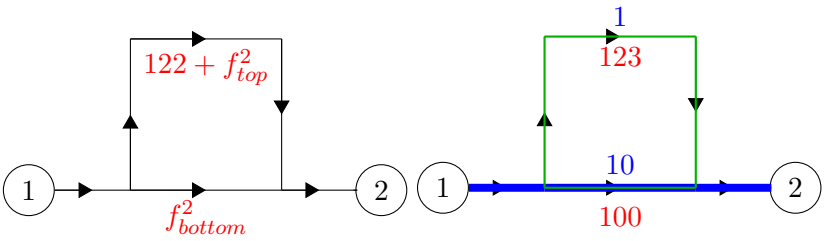
5.3 Junction modelling

In this section we will look at how junction modelling affects TAPAS in terms of number of PASs and in convergence rate.

5.3.1 Number of extra PASs

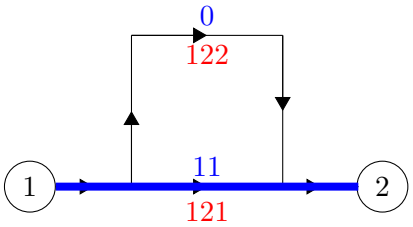
By expanding junctions we have added a lot of links and nodes, see Section 2.3, which may have an increase of PASs as a consequence. In this section we will look at the maximum number of extra PASs caused by junction expansion.

A PAS consists of two segment where we can change flow from one segment to the other and



(a) Network.

(b) Initial flow, where the green lines mark a PAS.



(c) Flow after one iteration.

Figure 5.2.5: Example where the total cost increases. The red formula near a link is the cost function, the red number the cost and the blue number the flow on that link. The blue link marks the minimum path between 1 *and* 2*. We have* $D_{1,2} = 11$*.*

still maintain conservation of flow. This means that a PAS consists of two paths with the same begin and end node. Suppose Figure 5.3.1 is part of our network and the blue and red lines mark the two segments of a PAS.

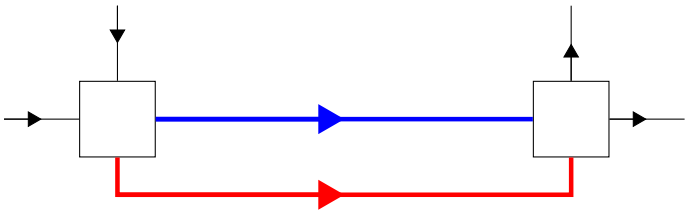


Figure 5.3.1: A PAS when there is no junction modelling.

In case of junction modelling the old nodes are replaced by the expanded junction, so they are replaced by multiple nodes and links (turns). We still require that the two segments of a PAS have the same begin and end node and have no other common nodes. In OmniTRANS there is maximum of 4 links involved in one junction for junction modelling. We will calculate now how many extra PASs are generated by expanding junctions when there are 4 links involved at every junction and one can go right, left and straight ahead. The old begin node is the left junction. One can leave this junction at the right side by arriving at this junction from the top, left or bottom side. One can leave this junction at the bottom side by arriving from the top, left or right side. When we want that one segment of the PAS leaves the junction at the right side and one segment at the bottom side and we want the same entrance point for both segments, we have two possible starting points: the top side and the left side. So

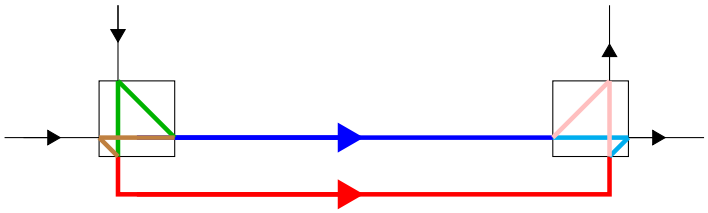


Figure 5.3.2: Four possible PASs are marked. Every PAS consists of the blue and red link and either the green or brown turns as begin point and either the pink or light blue turns as end point.

we have two possible starting points for a PAS where the old begin node was. The old begin node has now two possible begin points and, with similar reasoning, the old end node has now two possible end points, as can be seen in Figure 5.3.2. The maximum total amount of PASs in a network with junction modelling is therefore $2 \cdot 2 = 4$ times as many as in the network without junctions. When some turns are not allowed at specific junctions, the number of extra created PASs will be less than 4.
When the assumptions on the allowable number of links involved or the allowable turns are changed, the maximum number of extra PASs will change. The maximum number of extra created PASs can be determined in a similar way for other junctions. For example, we can see a roundabout as a junction where u-turns are allowed. When there are four links involved at the roundabout it will create a maximum of $4 \times 4 = 16$ times as many PASs as it had in a network without junction modelling.

5.3.2 Cost of a turn

There are many software packages that assume the cost of the turns to be constant during one or more iterations and are updated afterwards. One might expect a faster convergence when the turn cost formula is used. In some packages the formula of a turn contains multiple if-statements and is dependent on all turns in that junction, such that calculating turn costs takes a lot of time. This is the case for OmniTRANS. These complicated formula's make the calculation time for the cost of a turn high and make it difficult to determine if all types of junctions satisfy the conditions for a unique equilibrium. If the turn has a "simple" formula the formula can be used otherwise a (rough) estimation can be made or the cost can be held constant during one or more iterations. Our idea is that an estimation for the change in a turn cost can be made by means of the formula or by the changes caused by previous iterations, depending on the complexity of the formula.

5.3.3 Convergence

In this section we will look at the convergence of TAPAS with junction modelling. To determine why we can not use the proof of convergence of TAPAS (without junction modelling) one to one, we will first state the proof of convergence of TAPAS. Next we say something about the convergence of TAPAS in a network with junction modelling.

Convergence of TAPAS

This section start with proving two lemma's which are used to prove convergence of TAPAS in a network without junction modelling and next the convergence is proved. These proofs come almost word by word from the paper of TAPAS [6]. We start with making clear what an origin link combination (OLC) is by giving a very small example. In a network with 20 links and 4 origins, there are $4 \cdot 20 = 80$ OLCs.
The proof of convergence is based on the effectiveness of PAS shifts and branch shifts. What effective PASs and branch shifts are is explained in Section 5.2.1. We explained some modifications in Section 5.2, but the proof is still valid for TAPAS with these modifications since we still require effective PASs. It is sufficient to ensure that one of the two shifts is performed only for problematic OLCs, which are used OLCs (non-zero origin-based flow) with high reduced cost above a certain iteration-specific threshold, where the sequence of thresholds, Φ_k, converges to zero. We show first that there is a constant strictly positive lower bound on the reduction of the objective function in effective PAS shifts and in branch shifts. Since infinite repetitions of such reductions are not possible, this result is used to prove convergence by contradiction.
The function ξ in the following lemma's and theorem is defined as $\xi(f_{ij}^{r,\bullet}, rc_{ij}^{r,\bullet}) = \min(\eta^2 \cdot (rc_{ij}^{r,\bullet})^2/(8\kappa|A|), \eta \cdot \gamma \cdot f_{ij}^{r,\bullet} \cdot rc_{ij}^{r,\bullet}/2)$, where $f_{ij}^{r,\bullet}$ is the flow on link ij with origin r, $rc_{ij}^{r,\bullet} =$.

Lemma 1
The reduction in objective function of an effective PAS shift for OLC r, ij is at least $\xi(f_{ij}^{r,\bullet}, rc_{ij}^{r,\bullet}$, which is strictly positive if $f_{ij}^{r,\bullet} > 0$, $rc_{ij}^{r,\bullet} > 0$.
Proof
Denote the segment that ends at ij as segment 1, and the other segment by segment 2. Since the PAS is flow effective and cost effective the minimum flow along segment 1 is at least $\gamma \cdot f_{ij}^{r,\bullet}$, and the cost difference between segment 1 and segment 2 is at least $\eta \cdot rc_{ij}^{r,\bullet}$, with $0 \geq \gamma \geq 1$ and $0 \geq \eta \geq 1$, as described in Section 5.2.1. Recall that link cost functions are uniformly Lipschitz continuous with modulus κ, meaning that $|c_{ij}(f_{ij} + x) - c_{ij}(f_{ij})| \leq \kappa \cdot |x|$. Therefore, when an amount of flow x is shifted from segment 1 to segment 2, the change in the cost of every link in these segments is at most $\kappa \cdot x$. The total number of links in each segment is less than $|A|$, hence the total change in cost difference is at most $\kappa \cdot x \cdot 2|A|$. As long as the shift is less than $x \leq \eta \cdot rc_{ij}^{r,\bullet}/(4\kappa|A|)$, the cost difference remains at least $\eta \cdot rc_{ij}^{r,\bullet} - 2 \cdot x \cdot \kappa \cdot |A| \geq \eta \cdot rc_{ij}^{r,\bullet}/2$. If $\eta \cdot rc_{ij}^{r,\bullet}/(4\kappa|A|) \leq \gamma \cdot f_{ij}^{r,\bullet}$, meaning that a shift of $x = \eta \cdot rc_{ij}^{r,\bullet}/(4\kappa|A|)$ is feasible, the reduction in objective function will be at least $(\eta \cdot rc_{ij}^{r,\bullet}/2) \cdot (\eta \cdot rc_{ij}^{r,\bullet}/(4\kappa|A|)) = \eta^2 \cdot (rc_{ij}^{r,\bullet})^2/(8\kappa|A|)$. Otherwise, a shift of $x = \gamma \cdot f_{ij}^{r,\bullet}$ or more is implemented, which reduces the objective function by at least $(\gamma \cdot f_{ij}^{r,\bullet}) \cdot (\eta \cdot rc_{ij}^{r,\bullet}/2)$. In conclusion $\xi(f_{ij}^{r,\bullet}, rc_{ij}^{r,\bullet}) = \min(\eta^2 \cdot (rc_{ij}^{r,\bullet})^2/(8\kappa|A|), \eta \cdot \gamma \cdot f_{ij}^{r,\bullet} \cdot rc_{ij}^{r,\bullet}/2)$ satisfies the requirements of the lemma.

Lemma 2
The reduction in objective function of a branch shift for OLC r, ij is at least $\xi(f_{ij}^{r,\bullet}, rc_{ij}^{r,\bullet})$.
Proof
The cost of every segment in the branch from origin r that ends at link ij is at least $rc_{ij}^{r,\bullet}$ above the minimum cost alternative. Thus the difference between the average branch cost and the alternative segment is at least $rc_{ij}^{r,\bullet}$. As in the previous proof, a shift of x reduces the cost

difference by at most $\kappa \cdot x \cdot 2|A|$. If $x = rc_{ij}^{r,\bullet}/(4\kappa|A|) > f_{ij}^{r,\bullet}$ all the flow will be shifted, and the reduction in objective function will be at least $(f_{ij}^{r,\bullet}) \cdot (rc_{ij}^{r,\bullet}/2) \geq \xi(f_{ij}^{r,\bullet}, rc_{ij}^{r,\bullet})$. Otherwise, the amount of flow shifted is at least $x = rc_{ij}^{r,\bullet}/(4\kappa|A|)$ and the reduction in objective function is at least $(rc_{ij}^{r,\bullet})^2/(8\kappa|A|) \geq \xi(f_{ij}^{r,\bullet}, rc_{ij}^{r,\bullet})$.

Theorem 4
If $\Phi_n \to 0$ and in every iteration n for every problematic OLC with $f_{ij}^{r,\bullet} > 0$ and $rc_{ij}^{r,\bullet}(\mathbf{f}^{(n)}) > \Phi_n$ either effective PAS shift or a branch shift is applied, then the sequence of objective function values $O(\mathbf{f}^{(n)})$ converges to equilibrium.
Proof
The sequence $O(\mathbf{f}^{(n)})$ is monotonically non-increasing and bounded below by the optimal value of O. The set of feasible origin-based solutions is compact, therefore the sequence of solutions produced by the iterations has a converging subsequence $\mathbf{f}^{(n_l)} \to \tilde{\mathbf{f}}$. Suppose by contradiction that $\tilde{\mathbf{f}}$ is not an equilibrium solution, so that there exist origin r and link ij such that $\tilde{f}_{ij}^{r,\bullet} = x > 0$ and $rc_{ij}^{r,\bullet}(\tilde{\mathbf{f}}) = \epsilon > 0$. There exist l_0 such that $\Phi_{n_l} < \epsilon/2$, $f_{ij}^{(n_l)} > x/2$ and $rc_{ij}^{r,\bullet}(\mathbf{f}^{n_l}) > \epsilon/2$ for all $l > l_0$. Therefore, in each of these iterations the OLC r, ij is considered problematic, leading to either an effective PAS shift or a branch shift. In both cases the reduction in objective function is at least $\xi(x/2, \epsilon/2) > 0$. Infinite repetitions of such reduction leads to an infinite reduction in the objective function, which is a contradiction.

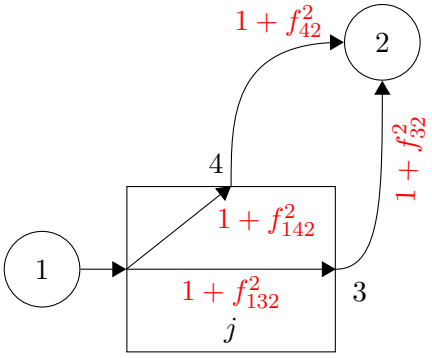
Convergence of TAPAS with junction modelling

We did not come up with a proof of convergence for modified TAPAS, but we stated our findings in this section. We can not use the proof of the convergence of TAPAS as described in Section 5.3.3, since there the fact that the cost of a link is only dependent on that link is used. The reduction in the objective value not only dependent anymore on the flow shifted and the reduction of the cost difference between the two segments of a PAS. This section is divided into 3 parts where the first part describes the convergence of TAPAS when the turn costs are assumed constant during the determination of the flow shift, in the second part we assume the turns in the segment are variable but the crossing turns are constant and in the last part we take all turns into account.

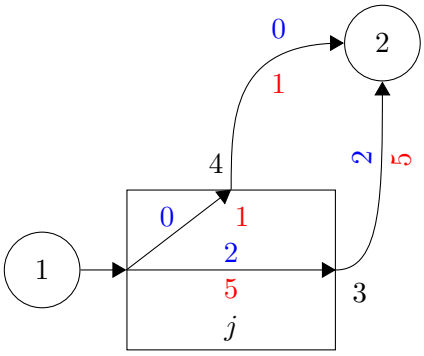
Constant turn costs

There are many software packages that assume the cost of the turns to be constant during the iteration, and are updated after a flow shift. In this situation convergence is not garanteed. Long calculation times caused by difficult formula's for turn costs are often the cause of the constant turn cost assumption. We will give an example of a network where the flow is shifted back and forth between the two routes when the turn costs are assumed to be constant during the determination of the amount of flow we want to shift.
Our example network is Figure 5.3.3a, where the demand is $D_{1,2} = 2$, the red formula's are the cost of the link/turn they are close to and the rectangle is junction j. Since we have not drawn any nodes between junction j and node 2 we will name the right side of the junction 3 and the top side 4. The turn from origin 1 via the top side of junction j to destination 2 is called 142 and has cost $c_{142} = 1 + f_{142}^2$. An AON-assignment gives Figure 5.3.3b, where the red numbers are the cost and the blue numbers the flow of that link. When we want to equalize the cost of both paths we try to minimize $5 + (1 + (f_{32} - x)^2) - (0 + 1 + (f_{42} + x)^2) =$

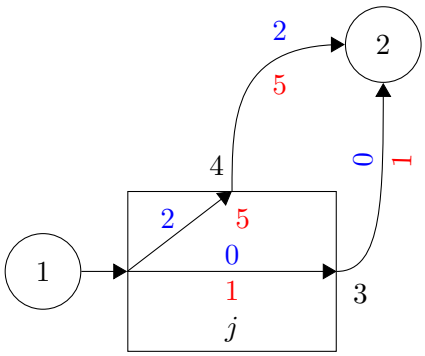
$(5 + 1 + (2 - x)^2) - (0 + 1 + x^2)$ where $0 \geq x \geq 2$. This results in a flow shift of 2, which is shifting all flow. The assignment we have after the flow shift can be found in Figure 5.3.3c. When we try again to equalize the cost of both paths we get the same answer: shifting 2 flow. This results in Figure 5.3.3b. The assignment will oscillate between (b) and (c) and will not converge to a UE. The UE assignment, which can be found if Figure 5.3.3d, will never be reached.
The turns are affecting the amount of flow we want to shift more when the turn is relatively expensive and increases/decreases much due to the flow shift.



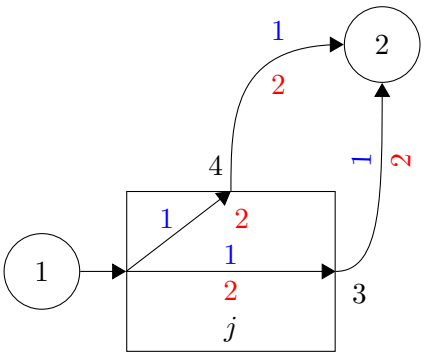
(a) Network where red formula nearby an arrow is the cost of that link or turn. The red number near link is the cost and the blue number is the flow of that link.



(b) Initial assignment.



(c) Assignment after 1 flow shift.



(d) UE assignment.

Figure 5.3.3: Example of network which does not converge if TAPAS is used and the turns are assumed to be constant during the iteration. $D_{1,2} = 2$.

Take only turns in segments into account

One can take only the turns into account which are part of the segments, and neglect the cost change in the other turns of the junctions. In this way not all affected turns have to be calculated during the iteration, saving a lot of time, and the costs of the segments can be equilibrated, where the costs of the segments will not change after the flow shift. The downside is that the objective function might increase due to the fact that we do not pay attention to the other turns. When the objective function might increase, we can not garantee

convergence. It can be that in real life networks this approach does not converge, but on the long term gets closer to equilibrium.

Take all affected turns into account

The third way of determining the amount of flow we want to shift between the two segments of a PAS is to take all affected turns into account. This takes a lot of time when the turn cost functions are complicated, but we are sure the objective function will not increase. This does not automatically mean that there is convergence, but we do not have a proof of convergence nor a counterexample. The proof of convergence for TAPAS for networks without junction modelling cannot be used one-to-one to this situation, since it is assumed that the cost of a link is only dependent on the flow of that link.
We cannot predict the decrease of the objective value by only looking at the flow on the two segments of a PAS and the reduced cost.

6 Results

We can see in the paper of TAPAS [6] that TAPAS should converge much faster than FW, and thus should converge faster than MSA. This result could not be duplicated by our Matlab-program. In this chapter we will explain what we did (not) do to get this result.

The network we used to test the algorithms in Matlab are grid networks with origin/destination nodes at the corner points where all links are two-way streets and every link has the same cost function and every turn has the same cost function. An example of a 2×2 grid network is Figure 6.0.1.

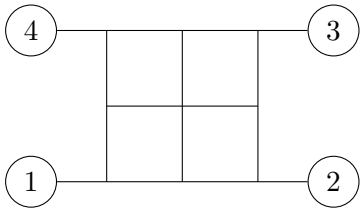


Figure 6.0.1: A 2×2 grid network. All links are two-way streets, all intersections of links are nodes/junctions and the circles are origins/destinations.

A grid network is in the Netherlands not a realistic network, which may affect the results. A symmetric grid may influence the convergence rate of MSA more than it does to TAPAS.

When we programmed TAPAS we had a lot of numbers as input and a lot of numbers as output. To better understand what happens we gave an output every iteration of the link cost and link flow. We made the thickness of a link dependent on the cost or flow on that link. Since all links have the same cost function, we get as output for the 5×5 grid network Figure 6.0.2, if the only positive demand is $D_{1,3} = 40$ (node at bottom left to node in top right).

The results of running our Matlab program on different raster networks can be found in Ta-

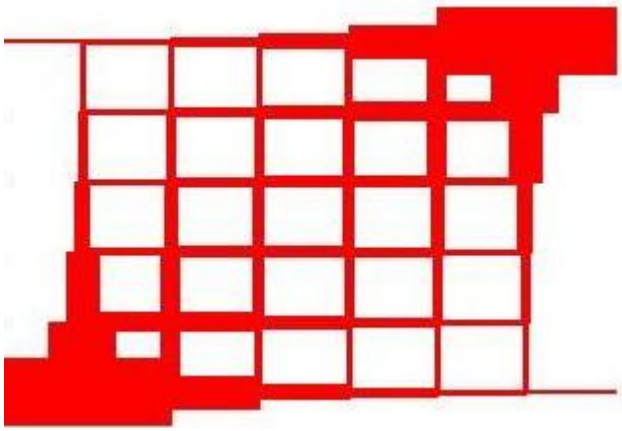


Figure 6.0.2: Matlab output of 5×5 network when $D_{1,3} = 40$.

Network	# OD-pairs	MSA		TAPAS	
		# iterations	time (sec)	# iterations	time (sec)
2×2	1	4	0.2	4	0.2
	12	2125	4.0	3	0.4
5×5	1	1284	7.9	10	8.3
	12	2256	13.9	8	12.4
10×10	1	3340	75.3	15	160.3
	12	3802	86.8	13	251.0
20×20	1	6412	866.8	22	6932.4
	12	6402	868.7	24	18723.9

Table 6.1: Results of running MSA and TAPAS in Matlab on an network without junction modelling.

bles 6.1, 6.2 and 6.3, where the first column in these tables stands for the size of the grid and the second column stands for the number of OD-pairs with positive demand where 1 means that there is only a positive demand for OD-pair 13, and 12 means that every possible OD-pair has demand (except OD-pairs 11, 22, 33 and 44). We did not draw the above mentioned figures when we determined these times.

When we run TAPAS with different random subsets the number of iterations and time may differ per run. To cancel out this effect we have done 3 runs for all network and written down the mean of these in the tables. Our stopping criterion was a relative gap (RGAP) of 10^{-4} or smaller. The time per iteration does not change much per run, but the number of iterations may vary. When we look at the 20×20 network with 12 OD-pair and no junction modelling, we got the following times: 33121.0 (37 iterations), 10475.2 (17 iterations) and 12575.4 (19 iterations). The mean is 18723.9, but the time varies much per run.

Table 6.1 is done on networks without any junction modelling. Here we can see that TAPAS needs more time than MSA on bigger networks, while in the paper TAPAS needed less time. Noticeble is that MSA needs many iterations where TAPAS only needs a few, but the time per iteration is longer for TAPAS.
Table 6.2 shows the results of the algorithms on networks with junction modelling. We have made the turn costs relatively low in comparison with the link costs in order to let the algorithm converge. Here we can see that TAPAS needs less time than MSA on a 5×5 grid network with 1 OD-pair, where the turn costs in MSA are updated every tenth iteration. When we look at Table 6.3, where the turn costs in MSA are updated every iteration, we see that MSA needs much more time than when it is updated every tenth iteration. This is caused by the fact that it takes a lot of time to calculate the turn costs. For TAPAS the turn cost is updated after every flow shift for all junctions which were affected by the shift.

We want to notice that how good an algorithm is, is not only dependent on the number of iterations, but on the combination of number of iterations and time per iteration.

Network	# OD-pairs	MSA		TAPAS	
		# iterations	time (sec)	# iterations	time (sec)
2×2	1	4	0.2	5	1.8
	12	188	20.6	6	5.5
5×5	1	1176	219.7	10	194.4
	12	2180	408.2	12	626.2

Table 6.2: Results of running MSA and TAPAS in Matlab on an network with junction modelling, where turn costs in MSA are updated every 10^{th} *iteration.*

Network	# OD-pairs	MSA	
		# iterations	time (sec)
2×2	1	162	10.4
	12	1881	119.1
5×5	1	938	1556.6
	12	2188	3659.4

Table 6.3: Results of running MSA in Matlab on an network with junction modelling, where turn costs are updated every iteration.

Due to time limitations we have not programmed a check for flow effectiveness or the possibility for branch shifts. We have to remark that it is checked that a PAS has flow, but it is not possible to take another value than zero for the γ in the flow-effective condition. This has a big effect on the convergence rate in certain networks, as explained in Chapter 5.2.1, but it is hard to predict the effect on our tests. Including a check for flow-effectiveness with $\gamma = 0$ improved the runtime.

Another reason we don't get the convergence rate of the paper is that the advantages of Matlab are not used. Matlab is very fast with matrix calculations, but less fast with for- and while-loops. We didn't do many matrix calculations and there are many loops in our Matlab-code, which causes a longer calculation time. We agreed on programming a proof of concept to show that TAPAS works, where it would be preferable if TAPAS converges faster than MSA. Since we programmed both algorithms in the same language we did initially not realize that the programming language makes that much of a difference. We have chosen Matlab since there is an interface for OmniTRANS in Matlab and we are more experienced in Matlab. The other option was using Ruby and C, since OmniTRANS is written in these languages. To make a good comparison between MSA and TAPAS one has to look critically at the programming code and at the (dis)advantages of programming languages.

We did not have enough time to optimize our Matlab program for speed. Some things one might change in order to get shorter runtimes is to combine some subprogrammes, like searching for a minimum cost tree and determining which links are used but are not part of this tree, change the cycle-find-program to the one mentioned in the paper or using another algorithm for searching shortest paths.

7 Conclusion and recommendations

In this chapter we will first state our conclusions and next the recommendations.

Conclusion

There exists a unique user equilibrium (UE) in terms of link-flow in a network without junction modelling, under the assumption that the cost function for every link is strict increasing and only dependent on the flow of that link.
We have proven that a network with junction modelling has a unique UE in terms of link-flow if $[c(x) - c(y)]^T(x - y) > 0$.

We have investigated a number of algorithms to find a UE. These algorithms are incremental assignment (IA), method of successive averages (MSA), Frank-Wolfe (FW), simplicial decomposition (SD), projected gradient method (PG), linear user cost equilibrium (LUCE), algorithm B (Alg. B) and traffic assignment by paired alternative segments (TAPAS). We have chosen TAPAS to look closer at since it converges fast and gives a proportional solution. It uses more memory than the algorithms used today in omniTRANS (IA, MSA, FW), but this is less and less an issue due to the growth in available computer memory.

The time needed to achieve the desired level of convergence in our Matlab program is relatively high compared to the runtimes mentioned in the paper [6]. This is caused by the fact that our program is not optimized for speed.
How and in which programming language the algorithm is programmed influences the runtime considerable. VA is more optimized for Matlab than TAPAS in our code, so comparing the two algorithms is difficult. To compare them more accurately one has to optimize both and for different networks instead of only on raster networks. In general, our tests showed that MSA needed less time than TAPAS to achive the desired level of convergence.

TAPAS was developed for networks without junction modelling. We developed three small modifications for TAPAS, but we have also investigated the changes in TAPAS when we add junction modelling and compared TAPAS with MSA on a network with juntion modelling. We can add junction modelling by expanding the junctions. When this is done for TAPAS the number of "pairs of alternative segments" (PASs) can increase to maximal 4 times the number of PASs in the network without junctions, when there is a maximum of 4 links involved at a junction and u-turns are prohibited.
The way in which junctions/turns are calculated also greatly affects the convergence time. The cost of a turn in OmniTRANS is complicated with multiple if-statements, making it too time consuming to determine which junctions satisfy the conditions for a unique UE, thus the cost of a turn is assumed to be constant during the iteration. This is also done in our tests. When the turn costs are updated every iteration, TAPAS needs less time than MSA to find a UE. When these costs are updated every tenth iteration, sometimes MSA is faster and sometimes TAPAS. As for networks without junction modelling, for a more accurately comparison one has to optimize the code of both algorithms and test them on different networks.

Recommendations

We recommend to investigate the convergence of TAPAS in networks with junction modelling, both theoretical and numerical.

We recommend to compare VA and TAPAS more in-depth for networks with junction modelling. One should consider which programming language is optimal and optimize both algorithms.

In TAPAS, one could investigate the effect of branch-shifts, since this was not included in our Matlab code.

During the determination of the flow we want to shift in a PAS in TAPAS we can decide to stop determining this flow and go on with the algorithm when certain values are smaller than a predetermined value, in this thesis they are called ϵ_2, ϵ_3 and ϵ_4. We recommend to investigate these ϵ_2, ϵ_3 and ϵ_4 into more detail, to determine how many PASs are deleted extra each iteration, the number of extra iterations needed and the increase or decrease of time per iteration. In other words, investigate the effect of the ϵ_2, ϵ_3 and ϵ_4 on the runtime.

References

[1] Omnitrans International. *OmniTRANS Manual, Compatible with OmniTRANS V5.1.16.*

[2] M. Mitradjieve-Daneva. *Feasible direction methods for constrained nonlinear optimization.* Linköping university, 2007.

[3] M. Florian, I. Constantin, and D. Florian. A new look at projected gradient method for equilibrium assignment. *Transportation research records*, 2090:10–16, 2009.

[4] G. Gentile. Linear user cost equilibrium: a new algorithm for traffic assignment. 2009.

[5] R.B. Dial. A path-based user equilibrium traffic assignment algorithm that obviates path storage and enumeration. *Transportation research part B*, 40:917–936, 2006.

[6] H. Bar-Gera. Traffic assignment by pared alternative segments. *Transportation Research Part B*, 44:1022–1046, 2010.

[7] J Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of civil engineers, part 2*, 1:325–378, 1952.

[8] M. Beckmann, C. McGuire, and C. Winsten. Studies in the economics of transportation. *Yale University Press*, 1956.

[9] J.B. Rosen. The gradient projection method for nonlinear programming, part 1: linear constraints. *Journal of the society for industrial and applied mathematics*, 8(1):181–217, 1960.

[10] L.H. Immers and J.E. Stada. Cursus h111, verkeersmodellen. Katholieke universiteit Leuven, 1998.

[11] W. Rudin. *Principles of mathematical analysis.* McGraw-Hill book company, 1986.

[12] K.C. Border. *Fixed point theorems with applications to economics and game theory.* Cabridge university press, 1985.

[13] M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval research logistics quarterly*, 3:95–110, 1956.

[14] H. Bar-Gera. Primal method for determining the most likely route flows oin a large road networks. *Transportation science*, 40:269–286, 2006.

A Adaptations on PG

We believe the paper about PG [3] defines variables in an incorrect way. We will explain in this chapter what we believe is incorrect, why we believe this and what we changed in order to use the algorithm in this thesis.

Suppose p and $\hat{p}$ are paths, $f_{\{p\}}$ is the flow on path p, $g_{\{p\}}$ is the descent direction of path p and λ^* is a scalar. $f_{ij}^{r,s}$ is the amount of flow on link ij with origin r and destination s. The cost of path p is $c_{\{p\}}$ and the average cost of all path from origin r to destination s is $\bar{c}_{r,s}$. δ_{ij}^{p} equals 1 if path p contains link ij and is 0 otherwise.

We update the flow of a path by $f_{\{p\}} = f_{\{p\}} + \lambda^* g_{\{p\}}$, where $0 \geq \lambda^* \geq \min\left\{ \frac{-f_{\{p\}}}{g_{\{p\}}} \middle| g_{\{p\}} < 0 \right\}$. The paper defines $g_{\{p\}} = c_{\{p\}} - \bar{c}_{r,s}$, where r and s are respectively the origin and destination of path p. A cheap path, that is a path with less cost than average, has $c_{\{p\}} < \bar{c}_{r,s}$, thus $g_{\{p\}} < 0$ in the paper. A cheap path has $\lambda^* \geq 0$ and $g_{\{p\}} < 0$, so $\lambda^* g_{\{p\}} < 0$. When updating the flow $f_{\{p\}} = f_{\{p\}} + \lambda^* g_{\{p\}}$ the paper decrease the flow on this path, while it was cheap. With similar reasoning the paper increase the flow of an expensive path. This leaves us with the conclusion that either $g_{\{p\}} = \bar{c}_{r,s} - c_{\{p\}}$ instead of $g_{\{p\}} = c_{\{p\}} - \bar{c}_{r,s}$ or $f_{\{p\}} = f_{\{p\}} - \lambda^* g_{\{p\}}$ instead of $f_{\{p\}} = f_{\{p\}} + \lambda^* g_{\{p\}}$.
To determine which one of these two we have to change, we will look at λ^*. The scalar λ^* should be bounded by an expensive path, since we extract flow from these paths. We have $0 \leq \lambda^* \leq \min\left\{ \frac{-f_{\{p\}}}{g_{\{p\}}} \middle| g_{\{p\}} < 0 \right\}$, thus $g_{\{p\}}$ should be an expensive path. If $g_{\{p\}} < 0$ is expensive, we get $g_{\{p\}} = \bar{c}_{r,s} - c_{\{p\}}$ instead of $g_{\{p\}} = c_{\{p\}} - \bar{c}_{r,s}$.

A second variable definition we want to point out is $G_{ij}^{r,s}$. The paper states this to be $G_{ij}^{r,s} = \sum_{p \in P_{r,s}^{+}} \delta_{ij}^{p} f_{\{p\}}$. We know $f_{\{p\}} \geq 0$ and $\delta_{ij}^{\{p\}}$ is either 1 or 0, leading to a non-negative $G_{ij}^{r,s}$. We want to find λ^* which minimizes $\min_{0 \geq \lambda \geq z} \sum_{ij \in A} \int_0^{f_{ij}^{r,s} + \bar{f}_{ij}^{r,s} + \lambda G_{ij}^{r,s}} c_{ij}(w) \mathrm{d}w$, where $\bar{f}_{ij}^{r,s} = \sum_{\substack{(r,s) \in D \\ r \neq s}} \sum_{p \in P_{r,s}^{+}} \delta_{ij}^{p} f_{\{p\}}$, $z = \min\left\{ \frac{-f_{\{p\}}}{g_{\{p\}}} \middle| g_{\{p\}} < 0 \right\}$ and w is a dummy variable for integration. All cost functions are increasing functions and the upper bound of the integrals can only be made greater by increasing λ, since $G_{ij}^{r,s} \geq 0$. Minimizing the above function will always lead to $\lambda^* = 0$, leaving us with the conclusion that $G_{ij}^{r,s}$ is defined incorrect. We will now derive the correct definition of $G_{ij}^{r,s}$. We know $f_{\{p\}}$ is updated by $f_{\{p\}} = f_{\{p\}} + \lambda^* g_{\{p\}}$. When we update $f_{ij}^{r,s}$, we get

$$\begin{aligned} f_{ij}^{r,s} = \sum_{p \in P_{r,s}^{+}} \delta_{ij}^{\{p\}} f_{\{p\}} &= \sum_{p \in P_{r,s}^{+}} \delta_{ij}^{\{p\}} \left(f_{\{p\}} + \lambda^* g_{\{p\}} \right) \\ &= \sum_{p \in P_{r,s}^{+}} \delta_{ij}^{\{p\}} f_{\{p\}} + \sum_{p \in P_{r,s}^{+}} \delta_{ij}^{\{p\}} \lambda^* g_{\{p\}} \\ &= f_{ij}^{r,s} + \lambda^* \sum_{p \in P_{r,s}^{+}} \delta_{ij}^{\{p\}} g_{\{p\}} \end{aligned}$$

We know $f_{ij}^{r,s}$ is updated by $f_{ij}^{r,s} + \lambda^* G_{ij}^{r,s}$, so we must conclude $G_{ij}^{r,s} = \sum_{\{p\} \in P_{r,s}^{+}} \delta_{ij}^{p} g_{\{p\}}$.
In this thesis we define $G_{ij}^{r,s} = \sum_{\{p\} \in P_{r,s}^{+}} \delta_{ij}^{p} g_{\{p\}}$ and $g_{\{p\}} = \bar{c}_{r,s} - c_{\{p\}}$.

B Adaptations on LUCE

Determining the bush in [4] is our vision not correct. We will show strange results when we use the definition of LUCE and thereafter state our definitions.

We will have a look at the notation first. We have the nodes i, $\tilde{\imath}$ and j. A link from i to j is called ij and the cost of this link is c_{ij}. The minimum cost path from i to destination s is C_i^s. The flow at link ij with destination s is f_{ij}^s and the alternative flow at link ij with destination s is e_{ij}^s. The bush of destination s is $B(^s)$.

The paper [4] states that the bush must be initialized by $B(^s) = \{ij \in A | C_i^s > C_j^s\}$, where the minimum cost are evaluated at zero flow. At the generic iteration the bush is modified by removing ij if $ij \in B(^s)$, $C_i^s < C_j^s$ and $f_{ij}^s = 0$ and adding ij if $ij \notin B(^s)$, $C_i^s > c_{ij} + C_j^s$ and $f_{ji}^s = 0$.

We begin with the generic iteration. We will explain here what we changed for which reason. Suppose the minimum path in a network is $i - \tilde{\imath} - j$, then $C_j^j = 0$, $C_{\tilde{\imath}}^j = c_{\tilde{\imath}j} + C_j^j = c_{\tilde{\imath}j}$ and $C_i^j = c_{i\tilde{\imath}} + C_{\tilde{\imath}}^j = c_{i\tilde{\imath}} + c_{\tilde{\imath}j}$. We have $C_i^j = c_{i\tilde{\imath}} + c_{\tilde{\imath}j} \not> c_{i\tilde{\imath}} + C_{\tilde{\imath}}^j = c_{i\tilde{\imath}} + c_{\tilde{\imath}j}$. Since the left side of the $\not>$-sign is not strictly greater than the right side (it's equal) the paper states link $i\tilde{\imath}$ should not be added to the bush. We believe that the minimum cost paths should be added to the bush, so we changed it to adding ij if $ij \notin B(^s)$ and $C_i^s \geq c_{ij} + C_j^s$.

The first iteration of LUCE on our example network is given at the next page, where the bush initialization is done as in [4]. This iteration results in $e_{12}^4 = \frac{0.4}{2\epsilon} + \frac{54}{10}$ and $e_{13}^4 = \frac{0.6}{3\epsilon} + \frac{54}{15}$. The alternative flow should be a number independent of ϵ. When we have $\epsilon = 0.0001$ we get $e_{12}^4 = \frac{0.4}{2 \cdot 0.0001} + \frac{54}{10} = 2005.4$, which is more than the total demand of the network. This is clearly a sign that the alternative flow is not computed correctly. From this we conclude that the bush must be initialized differently.

To stay close to the algorithm we used the adapted criterion of the generic iteration of adding a link: $ij \notin B(^s)$, $C_i^s \geq c_{ij} + C_j^s$ and $f_{ji}^s = 0$. Since all flow are zero, the last criterion is always fulfilled. We have $B(^s) = \{ij \in A | C_i^s \geq c_{ij} + C_j^s\}$ instead of $B(^s) = \{ij \in A | C_i^s > C_j^s\}$ for the bush initialization.

We will show the beginning of iteration 1 on our example network with an ordinary junction at node 3, when the bush is initialized as described in [4] here.

The network is Figure 4.6.1. We begin with iteration 1 and destination 4.
The costs are:
$c_{12} = 5 + 0^2 = 5$ $c_{13} = 11 + 2 \cdot 0^2 = 11$ $c_{23} = 5 + 0^2 = 5$
$c_{24} = 10 + 2 \cdot 0^2 = 10$ $c_{34} = 3 + 0^2 = 3$

and the derivatives:
$g_{12} = \max\{2 \cdot 0, \epsilon\} = \epsilon$ $g_{13} = \max\{2 \cdot 2 \cdot 0, \epsilon\} = \epsilon$ $g_{23} = \max\{2 \cdot 0, \epsilon\} = \epsilon$
$g_{24} = \max\{2 \cdot 2 \cdot 0, \epsilon\} = \epsilon$ $g_{34} = \max\{2 \cdot 0, \epsilon\} = \epsilon$

All flow is zero, so $y_{ij}^4 = 0 \; \forall ij \in A$.

The bush of destination 4, $B(^4)$, is the whole network, since we can see that $C_1^4 > C_2^4 > C_3^4 > C_4^4$.

$C_4^4 = 0,\ C_3^4 = 3,\ C_2^4 = 8,\ C_1^4 = 13$
$G_4^4 = 0,\ G_3^4 = \epsilon,\ G_2^4 = 2\epsilon,\ G_1^4 = 3\epsilon$

We set the alternative flow to zero
$e^4 = 0$.
Next, the demand is given
$e_1^4 = 9,\ e_2^4 = 2$.

We have to determine the alternative flow for every node in topological order, so $i = 1$
$J =$FSB$(1, 4) = \{2, 3\}$.

$$V_1^4 = \frac{9 + \dfrac{5+8}{3\epsilon} + \dfrac{11+3}{2\epsilon}}{\dfrac{1}{2\epsilon} + \dfrac{1}{3\epsilon}} = \frac{54\epsilon}{5} + \frac{68}{5}.$$

$$e_{12}^4 = \frac{0.4}{2\epsilon} + \frac{54}{10} \quad \text{and} \quad e_{13}^4 = \frac{0.6}{3\epsilon} + \frac{54}{15}$$

$e_{12}^4 > 0$ and $e_{13}^4 > 0$.

$$e_2^4 = e_{12}^4 = \frac{0.4}{2\epsilon} + \frac{54}{10} \quad \text{and} \quad e_3^4 = e_{13}^4 = \frac{0.6}{3\epsilon} + \frac{54}{15}.$$

We stop the iteration here, since e_{12}^4 and e_{13}^4 won't change during the rest of iteration 1 and it already shows the alternative flow to be incorrect.