LUCE: Linear User Cost Equilibrium

1 INTRODUCTION

In this chapter we present a new algorithm to solve the user equilibrium traffic assignment problem, called Linear User Cost Equilibrium (LUCE). The LUCE algorithm was conceived by Guido Gentile who during 2008 collaborated with PTV to produce a practical implementation of the method in VISUM. At the time of writing, shortly before the release of VISUM 11, the core method is stable enough to share it with our users, although some auxiliary functions are still missing and some post-assignment analysis methods still need to be optimized for LUCE. At this stage PTV provides LUCE as a prototype in VISUM 11. The prototype is provided mainly for evaluation purposes. It does run on realistic networks, but it currently has a few limitations, which will be lifted partly in VISUM 11 bugfixes, partly in the next major release, at which time LUCE will probably become the default equilibrium assignment method in VISUM.

The rest of the chapter contains a detailed description of the method (sections 2 – 5), followed by an explanation of practical usage within VISUM (section 6). Users who would like to get started quickly should read the very brief sketch of the method in this introduction, then skip to section 6.

Exploiting the inexpensive information provided by the derivatives of the arc costs with respect to arc flows, LUCE achieves a very high convergence speed, while it assigns the demand flow of each O-D pair on several paths at once.

Similarly to Origin-Based methods, the problem is partitioned by destinations. The main idea is to seek at each node a user equilibrium for the local route choice of drivers directed toward the destination among the arcs of its forward star. The travel alternatives that make up the local choice sets are the arcs that belong to the current bush – a bush is an acyclic sub-graph that connects each origin to the destination at hand. The cost functions associated to these alternatives express the average impendence to reach the destination linearized at the current flow pattern.

The unique solutions to such local linear equilibria in terms of destination flows, recursively applied for each node of the bush in topological order, provide a descent direction with respect to the classical sum-integral objective function. The network loading is then performed through such splitting rates, thus avoiding explicit path enumeration.

2 MATHEMATICAL FORMULATION AND THEORETICAL FRAMEWORK

The transport network is represented through a directed graph $G = (N, A)$, where N is the set of the nodes and $A \subseteq N \times N$ is the set of the arcs. The nodes include the zone centroids and the road intersections, while the arcs include the links and the connectors; when turns with impendence or restrictions are introduced in the network model, then the node is properly exploded, so that such turns are associated to specific or no arcs of the graph.

We adopt the following notation:

f_{ij} total flow on arc $ij \in A$, generic element of the $(|A| \times 1)$ vector **f** ;

c_{ij} cost of arc $ij \in A$, generic element of the $(|A| \times 1)$ vector **c** ;

$c_{ij}(f_{ij})$ cost function of arc $ij \in A$,

$Z \subseteq N$ set of the zone centroids ;

D_{od} demand flow between origin $o \in Z$ and destination $d \in Z$, generic element of the $(|Z|^2 \times 1)$ vector **D**, that is the o-d matrix in row major order ;

K_{id} set of the acyclic paths between node $i \in N$ and destination $d \in Z$;

$K = \cup_{o \in Z} \cup_{d \in Z} K_{od}$ is the set of paths available to users ;

δ_{ij}^{k} is 1, if arc $ij \in A$ belongs to path k, and 0, otherwise – for $k \in K$, this is the generic element of the $(|A| \times |K|)$ matrix $\boldsymbol{\Delta}$;

λ_{od}^{k} is 1, if path $k \in K$ connects origin $o \in Z$ to destination $d \in Z$ (i.e. $k \in K_{od}$), and 0, otherwise – this is the generic element of the $(|Z|^2 \times |K|)$ matrix $\boldsymbol{\Lambda}$;

F_k flow on path $k \in K$, generic element of the $(|K| \times 1)$ vector **F** ;

C_k the cost of path k – for $k \in K$ this is the generic element of the $(|K| \times 1)$ vector **C** ;

W_i^d the minimum cost to reach destination $d \in Z$ from node $i \in N$;

$\Re$ space of real numbers ;

$|S|$ cardinality of the generic set S ;

[TRUE] = 1, [FALSE] = 0 .

There are two fundamental relations between flow variables. The flow on arc $ij \in A$ is the sum of the flows on the paths that include it:

$f_{ij} = \sum_{k \in K} \delta_{ij}^{k} \cdot F_k$;

the travel demand between origin $o \in Z$ and destination $d \in Z$ must be equal to the sum of the flows on the paths that connect them:

$\sum_{k \in K_{od}} F_k = D_{od}$;

moreover, all path flows must satisfy non-negativity constraints.

As usual, we assume additive path costs, i.e. the impendence C_k associated by users to a given path k is the sum of the costs on the arcs that belong to it:

$$C_k = \sum_{ij \in A} \delta_{ij}^{k} \cdot c_{ij} \; . \tag{1}$$

By definition, the minimum cost to reach destination $d \in Z$ from node $i \in N$ is the cost of any shortest path that connects them:

$$W_i^d = \min\{C_k : k \in K_{id}\} \; . \tag{2}$$

In this case, the traffic assignment problem can be formalized through the following program:

$$\min\{\omega(\mathbf{f}) = \sum_{ij \in A} \int_0^{f_{ij}} c_{ij}(f) \cdot \mathrm{d}f : \mathbf{f} \in \Theta\}, \tag{3}$$

where:

$\Theta = \{\mathbf{f} \in \Re^{|A|} : \mathbf{f} = \mathbf{\Delta} \cdot \mathbf{F}, \mathbf{F} \in \Omega\}$ is the set of feasible arc flows, and

$\Omega = \{\mathbf{F} \in \Re^{|K|} : \mathbf{F} \geq \mathbf{0}, \mathbf{\Lambda} \cdot \mathbf{F} = \mathbf{D}\}$ is the set of feasible path flows.

To ensure the existence and uniqueness of the solution to problem (3) we assume that:

$c_{ij}(f_{ij})$ is non-negative, continuous, strictly monotone increasing ;

K_{od} is non-empty ;

D_{od} is non-negative .

Problem (3), which is convex, can also be expressed in terms of path flows as follows:

$$\min\{\Phi(\mathbf{F}) = \sum_{ij\in A} \int_0^{\sum_{k\in K} \delta_{ij}^{\,k} \cdot F_k} c_{ij}(f) \cdot \mathrm{d}f : \mathbf{F}\in\Omega\} \,, \tag{4}$$

where, although the solution uniqueness does not hold anymore, the convexity of the mathematical program is preserved, implying that any descent algorithm in the space of path flows will provide one of the global solutions, which then make up a convex set.

The relevance of program (4) to traffic assignment stands from the fact that, in the case of additive path costs, its first order (necessary) conditions coincide with the following formulation of the deterministic user equilibrium based on Wardrop's Principles, for each $o\in Z$ and $d\in Z$:

$$F_k \cdot (C_k - W_o^{\,d}) = 0 \,, \qquad \forall k\in K_{od} \,, \tag{5.1}$$

$$C_k \geq W_o^{\,d} \,, \qquad \forall k\in K_{od} \,, \tag{5.2}$$

$$F_k \geq 0 \,, \qquad \forall k\in K_{od} \,, \tag{5.3}$$

$$\sum_{k\in K_{od}} F_k = D_{od} \,. \tag{5.4}$$

Based on (5):

- all used paths ($F_k > 0$) have minimum cost ($C_k = W_o^{\,d}$) ;

- any unused path ($F_k = 0$) has not a lower cost ($C_k \geq W_o^{\,d}$) .

We have a user equilibrium if conditions (5) hold jointly for each o-d couple, while considering that each path cost C_k is a function (potentially) of all the path flows **F** through the arc cost function:

$$C_k = \sum_{ij\in A} \delta_{ij}^{\,k} \cdot c_{ij}\left(\sum_{k\in K} \delta_{ij}^{\,k} \cdot F_k\right) \text{ , in compact form } \mathbf{C} = \boldsymbol{\Delta}^{\mathrm{T}} \cdot c(\boldsymbol{\Delta}\cdot\mathbf{F}) \,. \tag{6}$$

Since the gradient of $\Phi(\mathbf{F})$ is $\mathbf{C} = \boldsymbol{\Delta}^{\mathrm{T}} \cdot c(\boldsymbol{\Delta}\cdot\mathbf{F})$, by linearizing the objective function of problem (4) at a given a point $\mathbf{F}\in\Omega$, for $\mathbf{X} \rightarrow \mathbf{F}$ we obtain:

$$\Phi(\mathbf{X}) = \Phi(\mathbf{F}) + \mathbf{C}^{\mathrm{T}} \cdot (\mathbf{X}-\mathbf{F}) + o(\|\mathbf{X}-\mathbf{F}\|). \tag{7}$$

From equation (7) we recognize that a direction **E-F** is descent if and only if:

$$\mathbf{C}^{\mathrm{T}} \cdot (\mathbf{E}-\mathbf{F}) < \mathbf{0}. \tag{8}$$

In other words, to decrease the objective function and maintain feasibility we necessarily have to *shift path flows getting a lower total cost with respect to the current cost pattern*, i.e. move the

current solution from **F** towards an $\mathbf{E}\in\Omega$, such that $\mathbf{C}^T\cdot\mathbf{E} < \mathbf{C}^T\cdot\mathbf{F}$, where $\mathbf{C} = \mathbf{\Delta}^T\cdot c(\mathbf{\Delta}\cdot\mathbf{F})$; the necessity derives from the convexity of the problem, since in this case at any point **X** such that $\mathbf{C}^T\cdot(\mathbf{X}-\mathbf{F}) > \mathbf{0}$ we have: $\Phi(\mathbf{X}) > \Phi(\mathbf{F})$.

This approach to determine a descent direction can be applied to each o-d pair separately, to each destination, or to the whole network jointly. Based on the above general rule, setting the flow pattern **E** by means of an all-or-nothing assignment to shortest paths clearly provides a descent direction. If we adopt such a direction for all o-d pairs of the network jointly, and apply along it a line search, we obtain the well known Frank-Wolfe algorithm. However, at equilibrium each o-d pair typically uses several paths, implying that any descent direction that loads a single path is intrinsically myopic; in fact such algorithms tail badly.

Once we get a feasible descent direction **E**-**F**, since Ω is convex, we can move the current solution along the segment $\mathbf{F}+\alpha\cdot(\mathbf{E}-\mathbf{F})$ and take a step $\alpha\in(0,1]$ such that the objective function of problem (4), redefined as $\phi(\alpha) = \Phi(\mathbf{F}+\alpha\cdot(\mathbf{E}-\mathbf{F}))$, is sufficiently lowered. In this respect, knowing that Φ is C^1 and convex, and thus also ϕ is such, several methods are available to determine an α which minimizes $\phi(\alpha)$. VISUM uses an Armijo-like search and determines the largest step $\alpha = 0.5^k$, for any non-negative integer *k*, such that

$$\partial\phi(0.5^k)/\partial\alpha < 0. \qquad (9)$$

This method requires to compute the directional derivative of the objective function:

$$\partial\phi(\alpha)/\partial\alpha = [c(\mathbf{\Delta}\cdot(\mathbf{F}+\alpha\cdot(\mathbf{E}-\mathbf{F})))]^T\cdot[\mathbf{\Delta}\cdot(\mathbf{E}-\mathbf{F})] , \qquad (10)$$

which implies to evaluate the arc costs at the candidate flows $\mathbf{F}+\alpha\cdot(\mathbf{E}-\mathbf{F})$, and then the difference between the corresponding total costs obtained with the flows **E** and **F**; if such total costs with **E** are smaller than those with **F**, then $\partial\phi(\alpha)/\partial\alpha$ is negative so that the optimal solution is more toward **E**, and vice versa.

3 LOCAL USER EQUILIBRIUM

In this section we present a new method to determine a descent direction, which is based on local shifts of flows that satisfy the total cost lowering rule and exploits the inexpensive information provided by the derivatives of the arc costs with respect to arc flows.

To grasp immediately the underlying idea, we can refer to the simplest network where one o-d pair with demand *D* is connected by two arcs with cost function $c_1(f_1)$ and $c_2(f_2)$, respectively. At the

current flow pattern $\mathbf{f}' = (D/2, D/2)$, it is $c_1' < c_2'$ (see Figure 1, below), so that an all or nothing approach would lead to a descent direction $(D, 0)$, which is far away from the equilibrium **f*** (grey circle in the Figure). The LUCE approach, instead, is to consider the first order approximations of the cost functions at the current flow pattern, i.e. $c_a' + \partial c_a(f_a)/\partial f_a \cdot (f_a - f_a')$, and determine a user equilibrium **e** among these lines (white circle in the Figure): this descent direction efficiently approaches the equilibrium **f***, and in most cases can be taken as the new iterate with a step one.

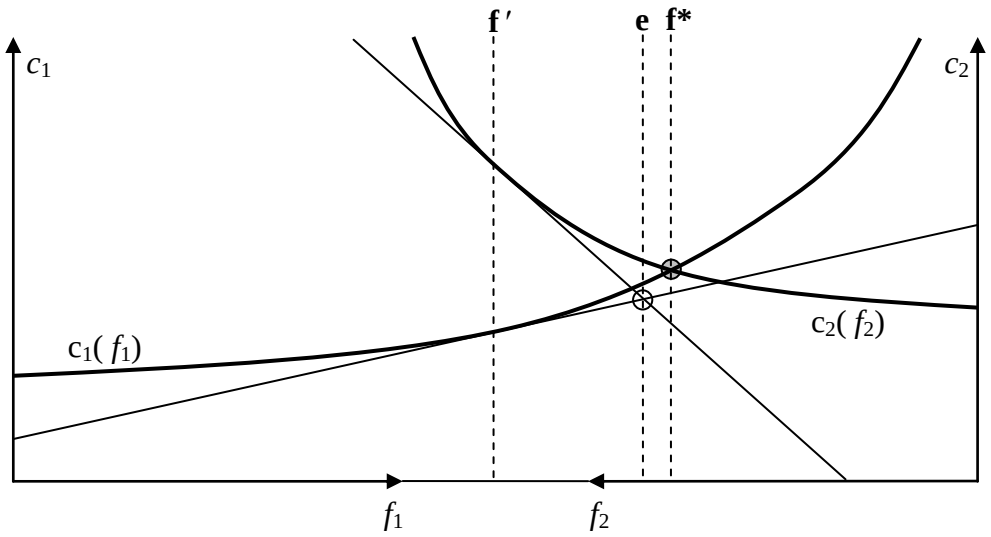


Figure 1. Linear Cost User Equilibrium between two paths.

To reach any destination $d \in Z$, at the equilibrium only shortest paths are utilized; given that the arc cost functions are strictly monotone increasing, they make up an acyclic [*1] sub-graph of G, i.e. a (reverse) bush rooted at d. On this base, when seeking a descent direction, in the following we will limit our attention to the current bush $B(d)$ and introduce a updating mechanism to make sure that eventually any shortest path will be included into it; only this way equilibrium is actually attained.

*[1] [In this case, indeed, any arc cost can be null only if its flow is such. However, in VISUM links and connectors may have null impedance, producing twofold consequences: a) the corresponding arc cost functions loose strict monotonicity, so that uniqueness is not guaranteed anymore; b) the sub-graph made-up by arcs with positive destination flows at some of the possible equilibria may be cyclic. The implementation of LUCE in VISUM specifically addresses this issue and converges to one among the possible equilibria by forcing an acyclic solution and equally splitting the flow

among all alternatives with minimum cost in presence of uncongested sub-paths.]

Let us focus on the local route choice at a generic node $i \in N$ for users directed to destination $d \in Z$.

For the topology of the bush we will use the following notation:

$FSB(i, d) = \{j \in N: ij \in B(d)\}$ the forward star of node $i \in N$ made-up by nodes that can be reached from it through arcs belonging to the current bush $B(d)$ of destination $d \in Z$;

$BSB(i, d) = \{j \in N: ji \in B(d)\}$ the backward star of node $i \in N$ made-up by nodes that can reach it through arcs belonging to the current bush $B(d)$ of destination $d \in Z$.

For the flow pattern we will use the following notation:

f_{ij}^d current flow on arc $ij \in A$ directed to destination $d \in Z$; by construction it is $f_{ij}^d = 0$ for each $j \notin FSB(i, d)$; moreover it clearly is: $f_{ij} = \sum_{d \in Z} f_{ij}^d$;

$f_i^d = \sum_{j \in FSB(i, d)} f_{ij}^d$ current flow leaving node $i \in N$ directed to destination $d \in Z$;

$y_{ij}^d = f_{ij}^d / f_i^d$ current flow proportion on arc $ij \in A$ directed to destination $d \in Z$, if $f_i^d > 0$; $y_{ij}^d = 0$, otherwise ;

e_{ij}^d descent direction, in terms of flow on arc $ij \in A$ directed to destination $d \in Z$;

e_i^d descent direction, in terms of flow leaving node $i \in N$ directed to destination $d \in Z$;

$x_{ij}^d = e_{ij}^d / e_i^d$ descent direction, in terms of flow proportion on arc $ij \in A$ directed to destination $d \in Z$.

For the cost pattern we will use the following notation:

C_i^d average cost to reach destination $d \in Z$ from node $i \in N$;

g_{ij} cost derivative of arc $ij \in A$;

G_i^d derivative of the average cost to reach destination $d \in Z$ from node $i \in N$.

The average cost C_i^d is the expected impendence that a user encounters by travelling from node $i \in N$ to destination $d \in N$; here it is defined recursively based on the current flow pattern:

if $f_i^d > 0$, then $C_i^d = \sum_{j \in FSB(i, d)} y_{ij}^d \cdot (c_{ij} + C_j^d)$, else (11.1)

$C_i^d = \min\{c_{ij} + C_j^d : j \in FSB(i, d)\}$, (11.2)

as if drivers utilize paths accordingly with the current flow proportions. In the following we assume that the cost function $c_{ij}(f_{ij})$ is continuously differentiable for each arc $ij \in A$:

$g_{ij} = \partial c_{ij}(f_{ij}) / \partial f_{ij}$. (12)

Under the assumption that an infinitesimal increment of flow leaving node $i \in N$ directed towards destination $d \in Z$ would diverge accordingly with the current flow proportions, we have :

if $f_i^d > 0$, then $G_i^d = \partial C_i^d / \partial f_i^d = \sum_{j \in FSB(i, d)} y_{ij}^{d\,2} \cdot (g_{ij} + G_j^d)$, else (13.1)

$G_i^d = \sum_{j \in FSB(i, d)} [C_i^d = c_{ij} + C_j^d] \cdot (g_{ij} + G_j^d) / \sum_{j \in FSB(i, d)} [C_i^d = c_{ij} + C_j^d]$, (13.2)

where the derivatives $g_{ij} + G_j^d$ are scaled by the share y_{ij}^d of ∂f_i^d utilizing arc ij and then passing through node j, that jointly with the flow proportion involved in the averaging yields the square $y_{ij}^{d\,2}$.

The average costs and their derivatives can be computed by processing the nodes of the bush in reverse topological order, starting from $C_d^d = G_d^d = 0$.

We now address the local user equilibrium for the e_i^d drivers directed to destination $d \in Z$, whose available alternatives are the arcs of the bush exiting from node $i \in N$. To each travel alternative we associate the cost function:

$v_{ij}^d(e_{ij}^d) = (c_{ij} + C_j^d) + (g_{ij} + G_j^d) \cdot (e_{ij}^d - y_{ij}^d \cdot e_i^d)$, (14)

resulting from a linearization at the current flow pattern of the average cost encountered by a user choosing the generic arc ij, with $j \in FSB(i, d)$.

This problem can be formulated, in analogy to (5), by the following system of inequalities:

$e_{ij}^d \cdot [v_{ij}^d(e_{ij}^d) - V_i^d] = 0$, $\forall j \in FSB(i, d)$, (15.1)

$v_{ij}^d(e_{ij}^d) \geq V_i^d$, $\forall j \in FSB(i, d)$, (15.2)

$e_{ij}^d \geq 0$, $\forall j \in FSB(i, d)$, (15.3)

$\sum_{j \in FSB(i, d)} e_{ij}^d = e_i^d$, (15.4)

where we denote:

V_i^d local equilibrium cost to reach destination $d \in Z$ from node $i \in N$;

v_{ij}^{d} cost of the local alternative $j \in FSB(i, d)$ to reach destination $d \in Z$ from node $i \in N$.

If $e_i^d = 0$, the solution to the above problem is trivially: $e_{ij}^d = 0$, for each $j \in FSB(i, d)$. Consider then the case where $e_i^d > 0$. To improve readability, problem (15) can be rewritten as:

$$x_j \cdot (a_j + b_j \cdot x_j - v) = 0 \ , \qquad \forall j \in J \ , \tag{16.1}$$

$$a_j + b_j \cdot x_j \geq v \ , \qquad \forall j \in J \ , \tag{16.2}$$

$$x_j \geq 0 \ , \qquad \forall j \in J \ , \tag{16.3}$$

$$\sum_{j \in J} x_j = 1 \ , \tag{16.4}$$

where:

$J = \{(i, j, d): j \in FSB(i, d)\}$;

$a_j = (c_{ij} + C_j^d) - (g_{ij} + G_j^d) \cdot e_i^d \cdot y_{ij}^d$;

$b_j = (g_{ij} + G_j^d) \cdot e_i^d$;

$x_j = e_{ij}^d / e_i^d$;

$v = V_i^d$.

Applying the usual Beckmann approach we can reformulate the equilibrium problem (16) as the following quadratic program:

$$\min\{\sum_{j \in J} {}_0\int^{x_j} (a_j + b_j \cdot x) \cdot \mathrm{d}x: \mathbf{x} \in X\} = \min\{\sum_{j \in J} a_j \cdot x_j + 0.5 \cdot b_j \cdot x_j^2: \mathbf{x} \in X\} \ , \tag{17}$$

where X is the convex set of all vectors satisfying the feasibility conditions (16.3) and (16.4). The gradient of the objective function is a vector with generic entry $a_j + b_j \cdot x_j$, and then the Hessian of the objective function is a diagonal matrix with generic entry b_j . Therefore, if all entries b_j are strictly positive, the Hessian is positive definite and problem (17) has a unique solution. In order to ensure such a desirable property we assume without loss of generality that the derivates g_{ij} are strictly positive for all arcs $ij \in A$. Indeed, since the arc cost functions are strictly monotone increasing, g_{ij} can be null only if also f_{ij}^d is null; therefore, at the equilibrium $b_j = 0$ implies $x_j = 0$. In practice we will substitute any $g_{ij} = 0$ with a small ε.

To solve problem (16) we propose the following simple method. In order to satisfy condition (16.1), either it is $x_j = 0$, and in this case condition (16.2) requires $a_j \geq v$, or it is $a_j + b_j \cdot x_j = v$. Let $J_0 \subset J$ be the set of alternatives with zero flow, that is $J_0 = \{ j \in J: x_j = 0\}$. For any given J_0 the solution is

immediate, since from (16.4) it is $\sum_{j \in J} (v - a_j) / b_j = 1$; therefore we have:

$$v = (1 + \sum_{j \in J \setminus J0} a_j / b_j) / (\sum_{j \in J \setminus J0} 1 / b_j) \ , \tag{18.1}$$

$$x_j = (v - a_j) / b_j \ , \qquad \forall j \in J \setminus J_0 \ , \tag{18.2}$$

$$x_j = 0 \ , \qquad \forall j \in J_0 \ . \tag{18.3}$$

The flow proportions provided by (18) implicitly satisfy (16.4), but to state that the chosen J_0 yields the solution of problem (16), we still must ensure the following conditions: $a_j < v$, for each $j \in J \setminus J_0$ (as required by (16.3), since $x_j = (v - a_j) / b_j > 0$), and $a_j \geq v$, for each $j \in J_0$ (as required by (16.2), since $x_j = 0$). This implies that at the solution the value of v resulting form (18.1) must partition the set J into two sub-sets: the set J_0 , made up by the alternatives j such that $a_j \geq v$; and its complement $J \setminus J_0$, made up by the alternatives j such that $a_j < v$.

At a first glance the problem to determine the set J_0 of alternatives with zero flow may seam to be combinatorial; however, this is not the case. Indeed, equation (18.1) can be rewritten as a recursive formula, thus showing the effect of removing an alternative k from the set J_0:

$$v[J_0 \setminus \{k\}] = (v[J_0] \cdot \sum_{j \in J \setminus J0} 1 / b_j + a_k / b_k) / (\sum_{j \in J \setminus J0} 1 / b_j + 1 / b_k) \ . \tag{19}$$

The right hand side of (19) can be interpreted as an average between $v[J_0]$ and a_k with positive weights $\sum_{j \in J \setminus J0} 1 / b_j$ and $1 / b_k$. Therefore, the local equilibrium cost increases by removing from J_0 any alternative $k \in J \setminus J_0$ for which a_k is higher than the current value $v[J_0]$, and vice versa it decreases by adding to J_0 such alternatives. Consequently, the correct partition set J_0 can be simply obtained by adding iteratively to an initially empty set each alternative $j \in J \setminus J_0$ such that $a_j > v$, i.e. each alternative for which (18.2) yields a negative flow proportion.

4 DESCENT DIRECTION

To obtain a complete pattern of arc flows $\mathbf{e}^d$ for a given destination $d \in Z$ consistent with the local user equilibrium we simply have to solve problem (15) at each node $i \in N \setminus \{d\}$ proceeding in topological order, where the node flow is computed as follows:

$$e_i^d = \sum_{j \in BSB(i,\, d)} e_{ji}^d + D_{id} \ . \tag{20}$$

In section 2 it has been shown that a given direction is descent if, and only if, (8) holds true, which in terms of arc flows directed to destination $d \in Z$ becomes:

$\sum_{ij \in A} c_{ij} \cdot (e_{ij}^{\,d} - f_{ij}^{\,d}) < 0$, (21)

expressing that the shift of flow from $\mathbf{f}^d$ to $\mathbf{e}^d$ must entail a decrease of total cost with respect to the current cost pattern. The proof that the proposed procedure provides a descent direction goes beyond the scope of this note and the interested reader is referred to the literature.

In the following we present an example showing the computation of the descent direction provided by the LUCE algorithm. We consider the graph of the Braess paradox, with 4 nodes and 5 arcs.

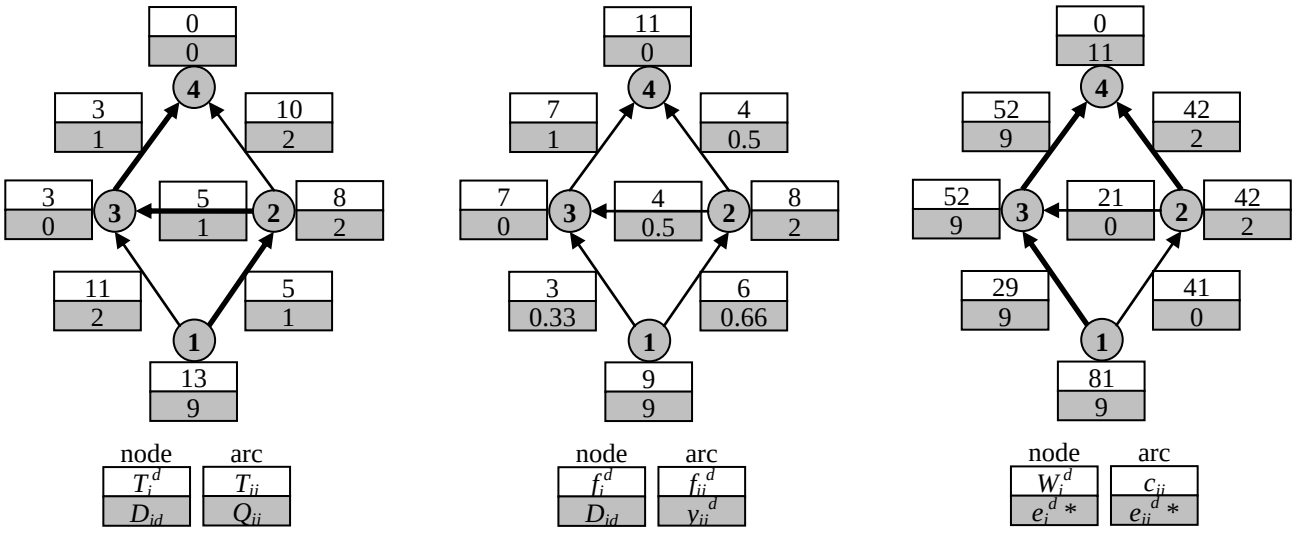


Figure 2A. Numerical example of the procedure to obtain the descent direction.

The arc cost function is $c_{ij} = T_{ij} + Q_{ij} \cdot f_{ij}^{\,2}$ so that its derivatives is $g_{ij} = 2 \cdot Q_{ij} \cdot f_{ij}$.

There is only one destination $d = 4$, and two origins with travel demand $D_{14} = 9$ and $D_{24} = 2$. We consider an initial flow pattern where all available paths, the 3 routes from 1 to 4 and the 2 routs from 2 to 4, are equally used by each o-d pair. Clearly, in this case it is $f_{ij} = f_{ij}^{\,d}$ and the bush is the entire network.

After we evaluate at the current flow pattern the arc costs and their derivatives, we can compute for each node i the average cost $C_i^{\,d}$ and its derivative $G_i^{\,d}$ iteratively stating from the destination, where $C_d^{\,d} = G_d^{\,d} = 0$, and proceeding in reverse topological order. To this aim we apply the formulas:

$C_i^{\,d} = \sum_{j \in FSB(i,\,d)} y_{ij}^{\,d} \cdot (c_{ij} + C_j^{\,d})$, $G_i^{\,d} = \sum_{j \in FSB(i,\,d)} y_{ij}^{\,d\,2} \cdot (g_{ij} + G_j^{\,d})$.

While the computation for node 3 is trivial, since its forward star is a singleton, for node 2 we have:

$C_2^4 = y_{23}^4 \cdot (c_{23} + C_3^4) + y_{24}^4 \cdot (c_{24} + C_4^4) = 0.5 \cdot (21 + 52) + 0.5 \cdot (42 + 0) = 57.5$,

$G_2^4 = y_{23}^{4\,2} \cdot (g_{23} + G_3^4) + y_{24}^{4\,2} \cdot (g_{24} + G_4^4) = 0.5^2 \cdot (8 + 14) + 0.5^2 \cdot (16 + 0) = 9.5$,

and for node 1 it is:

$C_3^4 = y_{13}^4 \cdot (c_{13} + C_3^4) + y_{12}^4 \cdot (c_{12} + C_2^4) = 0.33 \cdot (29 + 52) + 0.66 \cdot (41 + 57.5) = 92.7$,

$G_3^4 = y_{13}^{4\,2} \cdot (g_{12} + G_3^4) + y_{12}^{4\,2} \cdot (g_{12} + G_2^4) = 0.33^2 \cdot (12 + 14) + 0.66^2 \cdot (12 + 9.5) = 12.4$.

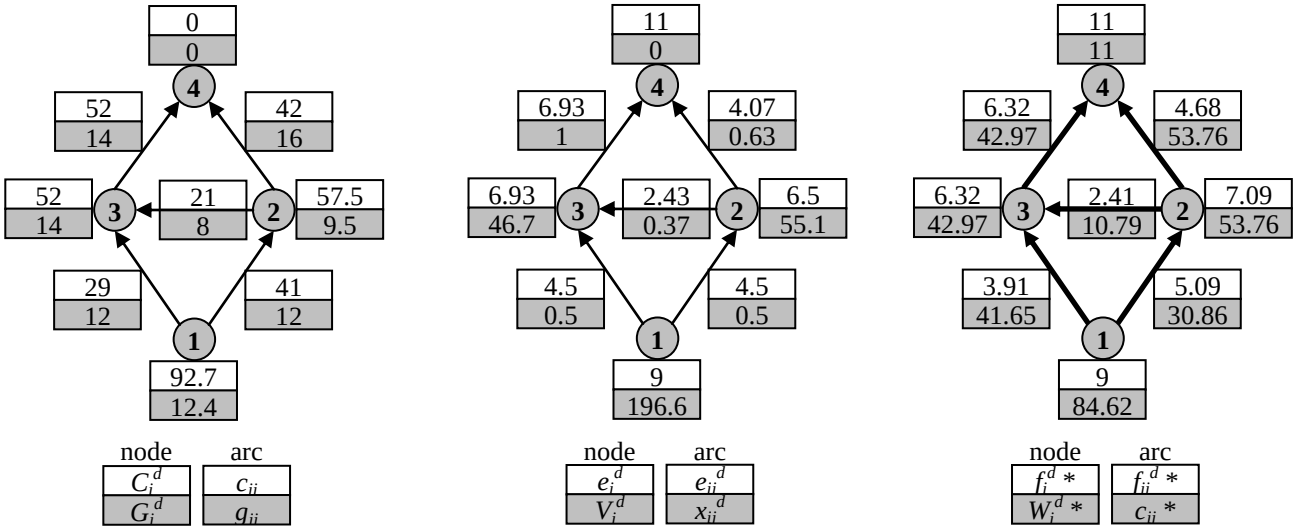


Figure 2B. Numerical example of the procedure to obtain the descent direction.

Now we can compute for each node i the node flows e_i^d and the arc flows e_{ij}^d iteratively by proceeding in topological order.

To this aim we shall focus on the local route choice of the e_i^d users, whose available alternatives are the arcs of the bush exiting from node i. To each travel alternative we associate the cost function:

$v_{ij}(e_{ij}^d) = (c_{ij} + C_j^d) + (g_{ij} + G_j^d) \cdot (e_{ij}^d - y_{ij}^d \cdot e_i^d)$,

resulting from a linearization at the current flow pattern of the average cost encountered by a user choosing arc ij, and we look for an equilibrium. We have shown that the latter can be determined using the following formulas:

$V_i^d = (1 + \sum_{j \in J} a_{ij}^d / b_{ij}^d) / (\sum_{j \in J} 1 / b_{ij}^d)$, $e_{ij}^d = e_i^d \cdot (V_i^d - a_{ij}^d) / b_{ij}^d$,

where: $a_{ij}^{d} = (c_{ij} + C_j^{d}) - (g_{ij} + G_j^{d}) \cdot e_i^{d} \cdot y_{ij}^{d}$, $b_{ij}^{d} = (g_{ij} + G_j^{d}) \cdot e_i^{d}$, while J is set initially to the forward star $FSB(i, d)$; if some e_{ij}^{d} results to be negative, then it is set to zero, j is removed from J and the computation is repeated.

We star then with node 1, whose node flow is $e_1^{4} = D_{14} = 6$:

$a_{13}^{4} = (c_{13} + C_3^{4}) - (g_{13} + G_3^{4}) \cdot e_1^{4} \cdot y_{13}^{4} = (29 + 52) - (12 + 14) \cdot 9 \cdot 0.33 = 3$,

$a_{12}^{4} = (c_{12} + C_2^{4}) - (g_{12} + G_2^{4}) \cdot e_1^{4} \cdot y_{12}^{4} = (41 + 57.5) - (12 + 9.5) \cdot 9 \cdot 0.66 = -30.5$,

$b_{13}^{4} = (g_{13} + G_3^{4}) \cdot e_1^{4} = (29 + 14) \cdot 9 = 387$,

$b_{12}^{4} = (g_{12} + G_2^{4}) \cdot e_1^{4} = (41 + 9.5) \cdot 9 = 454.5$,

$V_1^{4} = (1 + a_{13}^{4}/b_{13}^{4} + a_{12}^{4}/b_{12}^{4}) / (1/b_{13}^{4} + 1/b_{12}^{4}) = (1 + 3/387 - 30.5/454.5) / (1/387 + 1/454.5) = 196.6$,

$e_{13}^{4} = e_1^{4} \cdot (V_1^{4} - a_{13}^{4}) / b_{13}^{4} = 9 \cdot (196.6 - 3) / 387 = 4.5$,

$e_{12}^{4} = e_1^{4} \cdot (V_1^{4} - a_{12}^{4}) / b_{12}^{4} = 9 \cdot (196.6 + 30.5) / 454.5 = 4.5$.

Then we go on with node 2, whose node flow is $e_2^{4} = e_{12}^{4} + D_{24} = 4.50 + 2 = 6.5$:

$a_{23}^{4} = (c_{23} + C_3^{4}) - (g_{23} + G_3^{4}) \cdot e_2^{4} \cdot y_{23}^{4} = (21 + 52) - (8 + 14) \cdot 6.5 \cdot 0.5 = 1.5$,

$a_{24}^{4} = (c_{24} + C_4^{4}) - (g_{24} + G_4^{4}) \cdot e_2^{4} \cdot y_{24}^{4} = (42 + 0) - (16 + 0) \cdot 6.5 \cdot 0.5 = -10$,

$b_{23}^{4} = (g_{23} + G_3^{4}) \cdot e_1^{4} = (8 + 14) \cdot 6.5 = 143$,

$b_{24}^{4} = (g_{24} + G_4^{4}) \cdot e_1^{4} = (16 + 0) \cdot 6.5 = 104$,

$V_2^{4} = (1 + a_{23}^{4}/b_{23}^{4} + a_{24}^{4}/b_{24}^{4}) / (1/b_{23}^{4} + 1/b_{24}^{4}) = (1 + 1.5/143 - 10/104) / (1/143 + 1/104) = 55.1$,

$e_{23}^{4} = e_2^{4} \cdot (V_2^{4} - a_{23}^{4}) / b_{23}^{4} = 6.5 \cdot (55.1 - 1.5) / 143 = 2.43$,

$e_{24}^{4} = e_2^{4} \cdot (V_2^{4} - a_{24}^{4}) / b_{24}^{4} = 6.5 \cdot (55.1 + 10) / 104 = 4.07$.

Finally we consider node 3, whose node flow is $e_3^{4} = e_{13}^{4} + e_{23}^{4} + D_{34} = 4.5 + 2.43 + 0 = 6.93$:

Since there is only one alternative here, we have immediately $e_{34}^{4} = e_3^{4} = 6.93$, while we compute V_3^{4} only or completeness as follows:

$V_3^{4} = (c_{34} + C_4^{4}) + (g_{34} + G_4^{4}) \cdot (e_{34}^{4} - e_3^{4} \cdot y_{34}^{4}) = (52 + 0) + (14 + 0) \cdot (6.55 - 6.93 \cdot 1) = 46.7$.

The flow pattern we have just found is a descent direction because we have:

$\sum_{ij \in A} f_{ij}^{d} \cdot c_{ij} = 949 > \sum_{ij \in A} e_{ij}^{d} \cdot c_{ij} = 897$.

In Figure 2A, we have shown (denoted by an asterisk) the AON assignment on shortest paths. In Figure 2B, we show (denoted by an asterisk) the equilibrium flow and cost pattern. It can be seen that one single iteration of the proposed descent direction allows a substantial step towards the solution.

5 ASSIGNMENT ALGORITHM

Below we provide a pseudo code of the procedure within the framework of an assignment algorithm.

function LUCE

f = **0** initialize the solution flows to zero

for $k = 1$ **to** n perform n iterations

for each $d \in Z$ for each destination d

for each $ij \in A$ compute arc costs and their derivatives

$c_{ij} = c_{ij}(f_{ij})$

$g_{ij} = \max\{\partial c_{ij}(f_{ij})/\partial f_{ij}, \varepsilon\}$

if $f_i^d > 0$ **then** $y_{ij}^d = f_{ij}^d / f_i^d$ **else** $y_{ij}^d = 0$

$B(d) = B(B(d), \mathbf{c}, \mathbf{f})$ initialize or modify the current bush

$C_d^d = 0$ the average cost of the destination is zero

$G_d^d = 0$ so its derivative

for each $i: \exists ij \in B(d)$ **in reverse topological order** for each node $i \neq d$ in the bush

if $f_i^d > 0$ **then**

$C_i^d = \sum_{j \in FSB(i,d)} y_{ij}^d \cdot (c_{ij} + C_j^d)$ compute the node average cost to d

$G_i^d = \sum_{j \in FSB(i,d)} y_{ij}^{d\,2} \cdot (g_{ij} + G_j^d)$ and its derivative

else

$C_i^d = \min\{c_{ij} + C_j^d : j \in FSB(i,d)\}$

$G_i^d = \sum_{j \in FSB(i,d)} [C_i^d = c_{ij} + C_j^d] \cdot (g_{ij} + G_j^d) / \sum_{j \in FSB(i,d)} [C_i^d = c_{ij} + C_j^d]$

$\mathbf{e}^d = \mathbf{0}$ reset the arc and node flows to d

for each $o \in Z$ load on the origins the demand to d

$e_o^d = D_{od}$

for each $i: \exists ij \in B(d)$ **in topological order** for each node $i \neq d$ the bush

$J = FSB(i, d)$ initialize the set of arcs with positive flow

$\lambda = 0$

until $\lambda = 1$ **do**

$\lambda = 1$

$V_i^d = [e_i^d + \sum_{j \in J} (c_{ij} + C_j^d) / (g_{ij} + G_j^d) - e_i^d \cdot y_{ij}^d] / \sum_{j \in J} 1/(g_{ij} + G_j^d)$

for each $j \in J$

$e_{ij}^d = V_i^d / (g_{ij} + G_j^d) - (c_{ij} + C_j^d) / (g_{ij} + G_j^d) + e_i^d \cdot y_{ij}^d$

if $e_{ij}^d < 0$ **then**

$e_{ij}^d = 0$

$J = J \setminus \{j\}$ remove ij from the set of arcs with positive flow

$\lambda = 0$ then repeat the procedure

for each $j \in J$

$e_j^d = e_j^d + e_{ij}^d$ propagate the arc flows to the head node flows

$\alpha = 1$

if $k > 1$ **then**

until $\sum_{ij \in A} c_{ij}(f_{ij} + \alpha \cdot (e_{ij}^d - f_{ij}^d)) \cdot (e_{ij}^d - f_{ij}^d) < 0$ **do** $\alpha = 0.5 \cdot \alpha$ armijo step

for each $ij \in A$ update arc flows

$f_{ij} = f_{ij} + \alpha \cdot (e_{ij}^d - f_{ij}^d)$

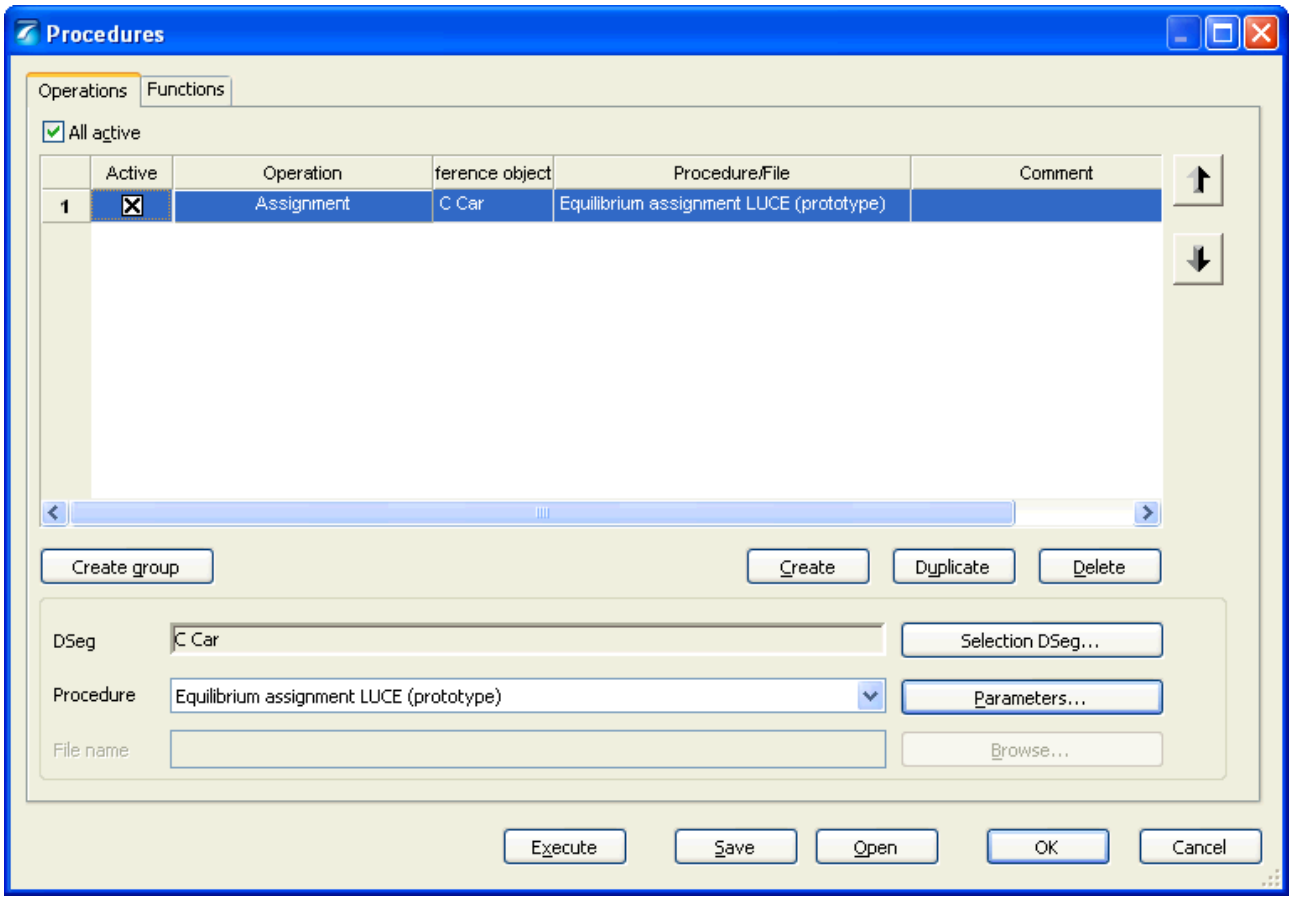
$f_{ij}^d = f_{ij}^d + \alpha \cdot (e_{ij}^d - f_{ij}^d)$

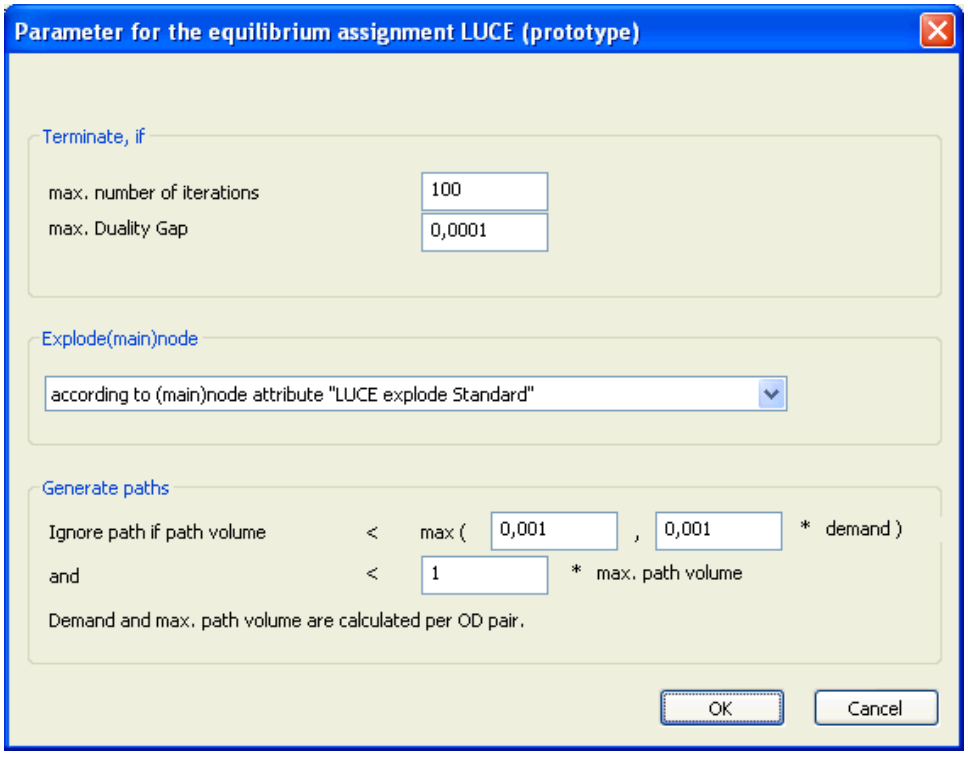
The bush of each destination $d \in Z$ is initialized with the set of efficient arcs that bring closer to the destination, where the minimum cost are evaluated at zero flow. At the generic iteration, any non-efficient arc on the bush carrying no destination flow is removed from it, while any arc that would improve shortest paths on the bush is added to it, if its reverse arc does not carry destination flow. If the resulting sub-graph is acyclic, then it is substituted to the current bush of that destination. Since the LUCE algorithm tends to an equilibrium on the bush, eventually the flow on non-efficient paths disappears and the bush can be properly modified.

Note that, beside the initialization of the bushes, the LUCE algorithm does not require shortest path computations, but only simple visits of the bushes.

6 USAGE

LUCE is one of the private transport assignment methods in VISUM 11. To run a LUCE assignment set up a model run under *Calculate – Procedures* and select *Equilibrium assignment LUCE* as the sub-method of operation Assignment.





Set the convergence criteria in the *Terminate, if* section.

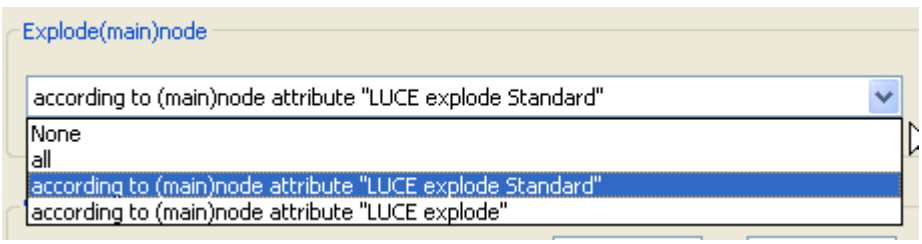
The other procedure parameters control memory consumption and route storage. LUCE differs from all other VISUM assignment methods in that it is not explicitly path-based, but uses an implicit representation of the loaded paths, the bushes explained above. The main benefit of this representation is that LUCE can load a richer set of paths than VISUM's classic path-based equilibrium assignment, in a limited number of iterations. The representation has an important effect on memory consumption:

Memory consumption for …	**classic equilibrium assignment**	**LUCE**
path storage	explicit storage of all paths	more compact implicit storage of bushes
link / turn / connector volumes	one volume for each network element per demand segment	one volume for each network element per demand segment **and origin**

The table shows that path storage is more memory-efficient in LUCE, but network volumes need to be stored per origin zone which can consume much more memory than in the classic assignment. To minimize this effect it is important to store volumes only for those network elements which are essential in route choice. Link and connector volumes are always stored. (Main) turns, which contribute the largest number of network elements, are only relevant to the assignment, if they differ in impedance. Basically, if the turns at a given node have identical transport system sets and identical delays, then their impedances and volumes can be ignored in the calculation, and memory saved. If not, the node needs to be "exploded", i.e. the turns need to be added as arcs to the graph for the assignment, as explained in section 2 above.

Exploding nodes

You control the explosion of nodes through the *Explode (main) node* parameter:



There are four possible values:

none: (main) turns are generally not exploded, i.e. differences in impedance (if present) are ignored

all: (main) turns are exploded at all nodes

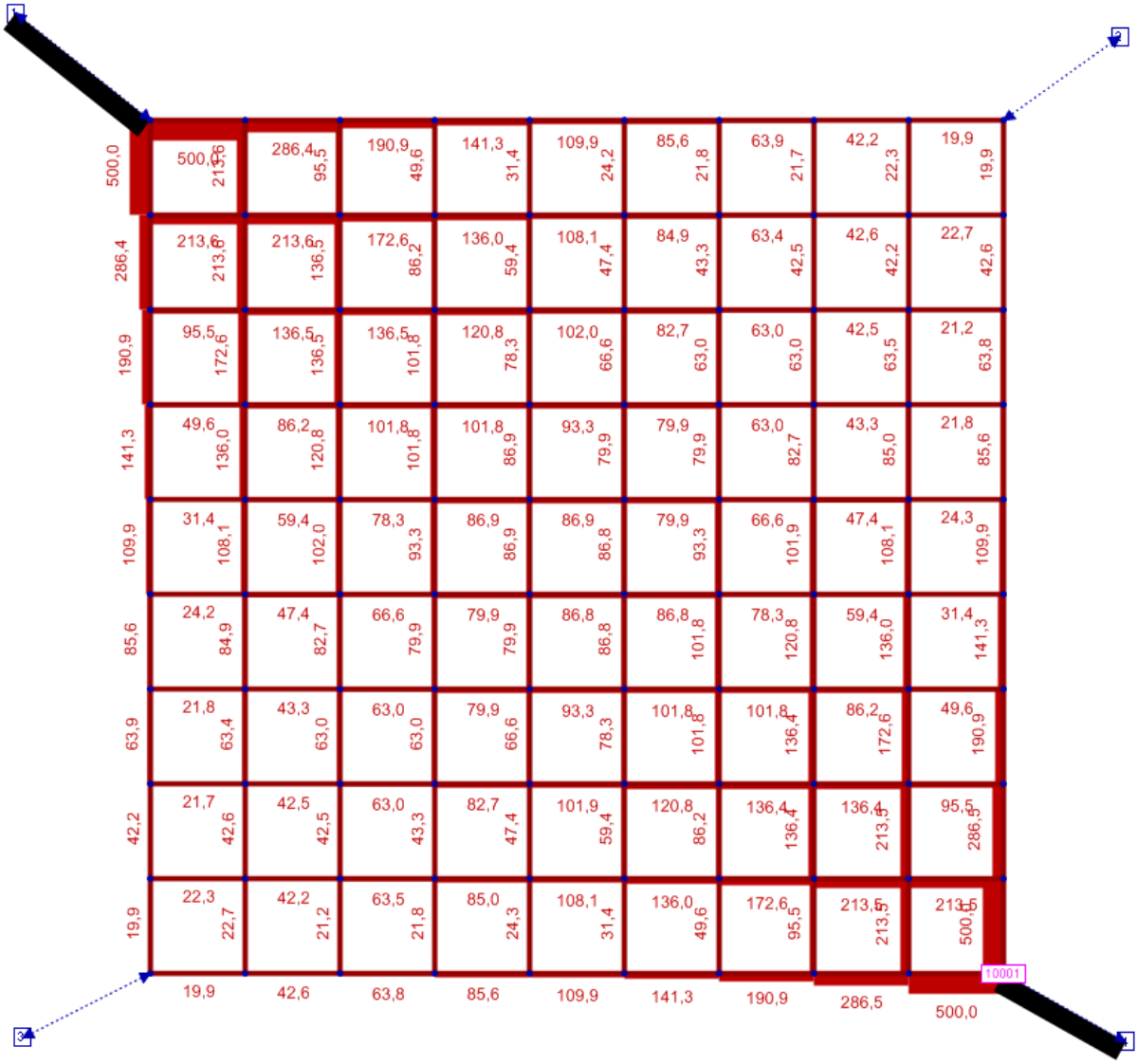
according to (main) node attribute "LUCE explode Standard" (default): VISUM pre-calculates the zero-or-one attribute *LUCE explode Standard*. The value is 1, if the turns differ and need to be exploded, and 0, if they can be safely ignored.

according to (main) node attribute "LUCE explode": While *LUCE explode Standard* is a read-only attribute, *LUCE explode* is editable. If you need fine control over which nodes are exploded, copy the values of *LUCE explode Standard* to *LUCE explode*, and modify them where necessary.

In most cases it is best to accept the default. Choose the second or fourth option, if you plan to warm-start LUCE subsequently, and for the new run (main) turn impedances or transportation system sets will be different from the first run. Any (main) turns to be exploded in the second run must already be exploded in the stored result of the first run for warm-start to work. Choose *all*, if you can afford the extra memory consumption, or choose the fourth option and set the *LUCE explode* attribute to 1 for exactly those (main) nodes that should be exploded. Use the option *none* for a fast sketch-level assignment ignoring turn impedances altogether.

Route extraction

LUCE's big advantage over VISUM's classic assignment algorithm is the richer path sets it loads. The classic algorithm will load at most one path per O-D pair and iteration. Consider the totally symmetric grid graph displayed below to which traffic is assigned for a single O-D pair (top left to bottom right).

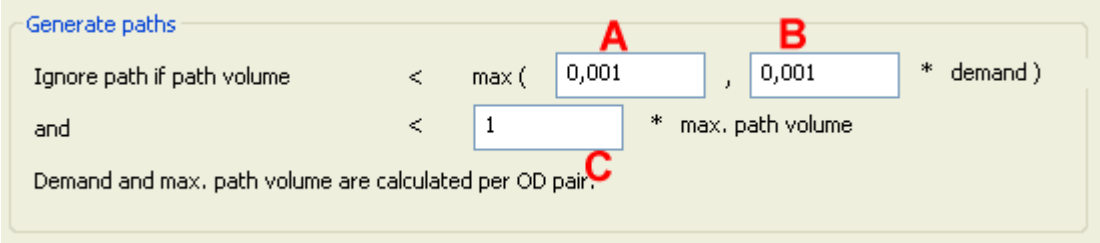


If we run the classic algorithm on this graph for 100 iterations, exactly 100 paths will be found and loaded in the final result. VISUM will reach the unique equilibrium link volumes with a very good gap. Note, however, that unlike link volumes, route flows are not unique in equilibrium assignment. The 100 loaded paths clearly represent an extreme corner solution in the space of route flow patterns which are consistent with the link volumes. In contrast, LUCE loads over 34000 paths in just 85 iterations.

The artificial example demonstrates that in networks with many attractive alternative routes, the number of loaded paths per O-D pair can be quite high. The assignment algorithm itself does not work with paths, and in some applications it is never necessary to extract the loaded routes, e.g. if link volumes are sufficient. But for various post-assignment operations, e.g. select-link analysis or

matrix estimation, routes are necessary.

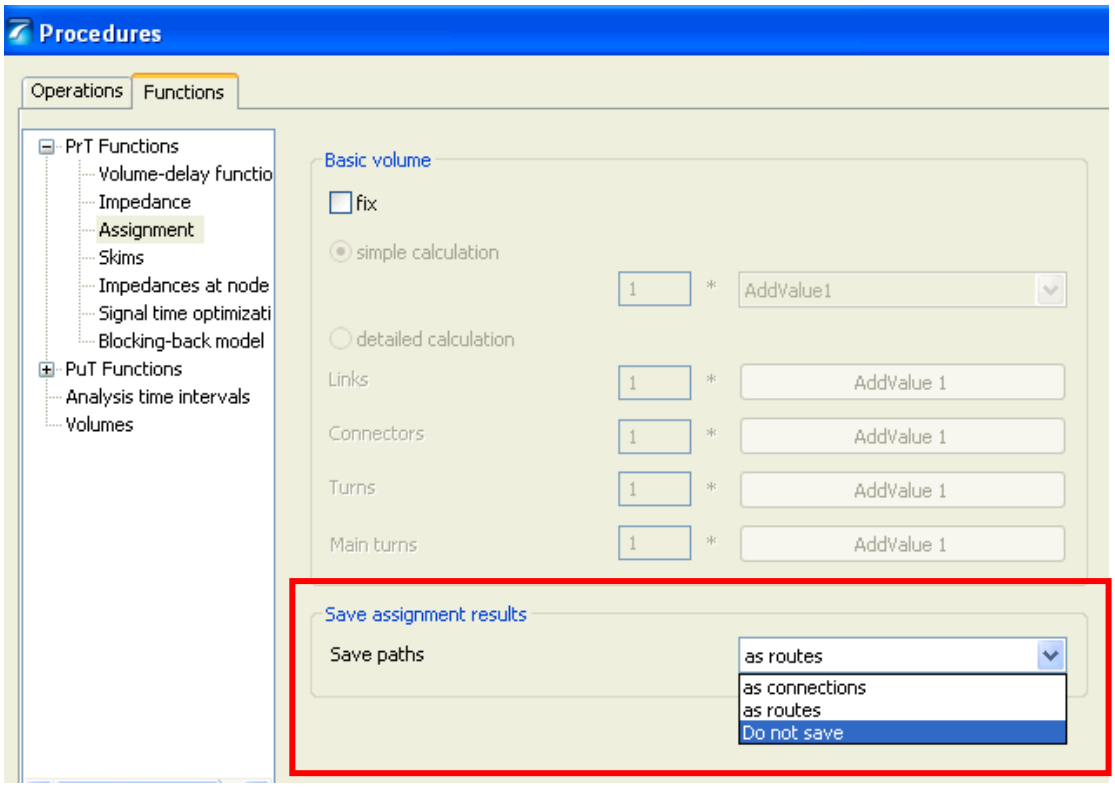
Extracting *all* routes from the compact bush representation may be impossible, for memory reasons, in large, highly symmetric networks. In such extreme cases, most of the routes will have exceedingly small volumes. In order to avoid being swamped by an astronomical number of very-low-volume links, VISUM lets you control the set of paths to be extracted:



Use the three parameters to define a cut-off point below which paths are not extracted. Any demand on the ignored paths is proportionately redistributed to the surviving paths.

Parameter A defines an absolute cut-off point. For O-D pairs with a high number of trips the absolute cut-off point may still be inconveniently small and too many paths survive. Use parameter B in these cases to define a cut-off point relative to the total demand of the O-D pair. In some cases, there may be many paths for a single O-D pairs, but all of them have very small volumes. Then there is a danger that all paths fall below the cut-off point defined by A and B. Parameter C then ensures that at least the paths with (near-) maximum volume will always survive.

For those cases where you do not need routes at all, you can save memory by turning off route extraction altogether.



Under *Calculate – Procedures – PrT Functions – Assignment* choose the *Do not save* option for paths, in which case LUCE will only save link volumes, but no paths at all. The default setting is *Save Paths as routes*, which stores paths. The third option (*as connections*) is reserved for dynamic assignment methods.

7 PERSPECTIVES

The LUCE algorithm released with VISUM 11 is fully functional. Some extensions are already planned and – where possible – will be added even before the next major release:

Warm start: Like other assignment methods in VISUM, LUCE will be able to accept an existing assignment result as an initial solution. Because the warm start functionality requires a set of bushes, the prior assignment result must be of type LUCE.

Faster skim matrices and select-link analysis (flow bundle): In the initial release, all post-assignment analysis functions are available for LUCE, because from the bush representation of the equilibrium solution VISUM extracts paths in the classic format. This can be a memory bottleneck in some networks (see above). Some of the analysis methods can actually be re-implemented to work directly with the implicit bush representation which not only saves memory, but also speeds up the operation. Bush adaptations of these methods will be added to VISUM at a later time.

Bush storage: Similar to route extraction, LUCE will gain an option to enable / disable bush storage and save memory, because bushes only need to be saved, if you plan to warm-start or use post-assignment analysis.

General tuning: Although we have tested LUCE on diverse networks we fully expect performance (in terms of memory and runtime) to vary with the characteristics of the networks, and it is quite likely that we will need to tune the implementation for the cases that have escaped us so far. You can help us by reporting to the VISUM hotline instances in which LUCE consumes unlikely amounts of runtime or memory.