Demand Modelling with EMME/2

Presented by

Isabelle Constantin, INRO Consultants Inc.

2nd EMME/2 Asian Users' Group Meeting
– Hong Kong, November 30 – December 1, 2000

File: e2dmodHK

Tools Available in EMME/2

Instead of specific demand models, EMME/2 provides **tools**:

- Matrix manipulation tools
  - matrix **calculations**
  - matrix **balancing**
  - **matrix product** like operations

- Assignment procedures
  - equilibrium **auto assignment** (multiclass, generalized cost)
  - multi-path **transit assignment**

- **Macro** language

⇒ Allows implementing a **wide variety** of demand models

Implementing a 4-Step Model with EMME/2

Land use and socio-economic data

Trip generation/attraction — Module 3.21

Productions and attractions

Trip distribution — Module 3.21 & 3.22 (3.16)

Demand by O-D pair

Mode choice — Module 3.21

Demand by O-D pair by mode

Assignment — Module 5.11, 5.21 & 5.31

Travel times and costs by O-D pair
Volumes on network

Trip Generation & Attraction Models

- Zone data:
  - **population characteristics**: places of residence, dwelling types, age, …
  - **economic activities**: number of jobs, type of industry, office surface area, …

- Models:

  attraction and production coefficients obtained by calibrating models (econometric models, category analysis, …)

- Examples:

$$O_p = \boldsymbol{F}(household\_size_p,\ household\_income_p)$$

"produc"	**="homes"*2.9**	for suburban zones
	="homes"*1.7	for center zones

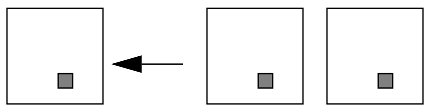
$$D_q = \boldsymbol{F}(retail\_employment_q,\ other\_employment_q,\ households_q)$$

"attrac"	**="indust"*350+"office"*40**	for industrial zones
	="indust"* 75+"office"*80	for business zones

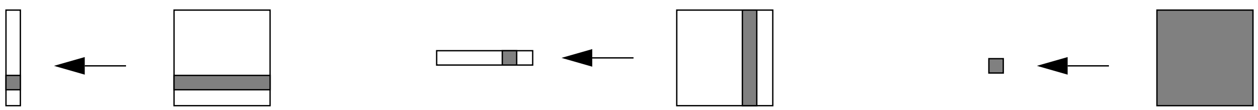
Matrix Calculations (Module 3.21)

- Evaluation of algebraic expressions involving matrices or zone data

- Calculation performed element by element: each element r_{pq} of result matrix is obtained from the elements *pq* of the operand matrices



- Aggregation performed for vector or scalar results:



- Useful for calculations performed for origins and/or destinations, including **many kinds of demand models**

Trip Distribution Models

- Aim: **reproducing productions and attractions** (observed or predicted)
- Different input matrices yield different models. Examples:

Growth model (also called **scaling** or **Fratar**)

- starts with an existing O-D matrix

Gravity model

- minimize total travel time or cost.
- starts with a function of the travel time or cost $f_{pq} = F(u_{pq})$
- sometimes specified as a **look-up table**

Two-Dimensional Matrix Balancing (Module 3.22)

- Aim: determine an origin-destination matrix, g_{pq}, satisfying:

$$\sum_q g_{pq} = O_p \qquad \text{for each } p \qquad \textit{(productions are respected)}$$

$$\sum_p g_{pq} = D_q \qquad \text{for each } q \qquad \textit{(attractions are respected)}$$

$$g_{pq} \geq 0 \qquad \text{for each } (p, q)$$

- The resulting **balanced matrix** can be expressed as $g_{pq} = \alpha_p \beta_q f_{pq}$
where: α_p and β_q are the **balancing coefficients.**

- Useful for implementing distribution models:
different input matrices $f_{pq} \Rightarrow$ different models.

2-D Matrix Balancing Algorithm

Step 0. Initialization

$$l = 0$$

$$\alpha_p^{(0)} = 1 \quad \text{for each } p$$

$$\beta_q^{(0)} = 1 \quad \text{for each } q$$

Step 1. Balancing rows

$$\alpha_p^{(l+1)} = \frac{O_p}{\sum_q \beta_q^{(l)} f_{pq}} \quad \text{for each } p$$

Step 2. Balancing columns

$$\beta_q^{(l+1)} = \frac{D_q}{\sum_p \alpha_p^{(l+1)} f_{pq}} \quad \text{for each } q$$

Step 3. Stopping test

$$l = l + 1$$

If max $\left| \alpha_p^{(l)} \cdot \sum_q \beta_q^{(l)} f_{pq} - O_p \right| \leq \varepsilon$ **stop**

else return to **Step 1**.

Note: If $\sum_p O_p = \sum_q D_q$, this algorithm converges (in most cases)

Example: Gravity Model Based on Look-up Table

- Prepare input matrix according to look-up table

Interval	Travel time	F-factor
1	0-5	0.15
2	5-10	0.22
3	10-15	0.27
4	15-20	0.24
…		

Example of EMME/2 macro (for module 3.21)

```
~g1=.15
~g2=.22
~g3=.27
~g4=.24
…
"gravit" = get(int("time"/5)+1)*"ffact"
```

- 2-dimensional balancing (module 3.22)
  - specify productions and attractions
  - input matrix: **`"gravit"`**

Example: Entropy Model

- Special case of gravity model:
  - maximize entropy (small values everywhere instead of large values and 0’s)
  - minimize total travel time (or cost) ⇨ $f_{pq} = e^{-\theta u_{pq}}$

    u_{pq} = estimated cost

    θ = constant to calibrate

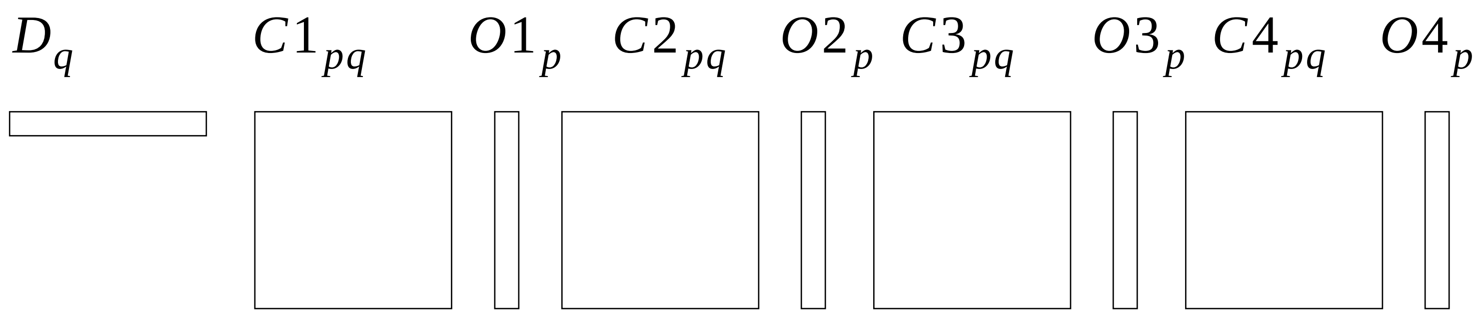
- Prepare input matrix (module 3.21)

  ```
  "entrop" = exp(-.059*"autime")*("obsdem">0)
  ```

- 2-dimensional balancing (module 3.22)
  - specify productions and attractions
  - input matrix: **`"entrop"`**

2-D Balancing With Multiple Production Types: BALMPROD

- One attraction vector D_q and up to 4 production vectors Oi_p and corresponding friction matrices Ci_{pq} can be specified:



Must satisfy $\sum_q D_q = \sum_p O1_p + \sum_p O2_p + \sum_p O3_p + \sum_p O4_p$

- Results:
  - balancing coefficients by destination: **`md91`**
  - balancing coefficients by origin for production type *i*: **`mo9`***i*
  - the balanced matrix for production type *i* can then be computed as:
    `mo9`*i* * **`md91`** * **`mf"Cpq`***i***`"`**

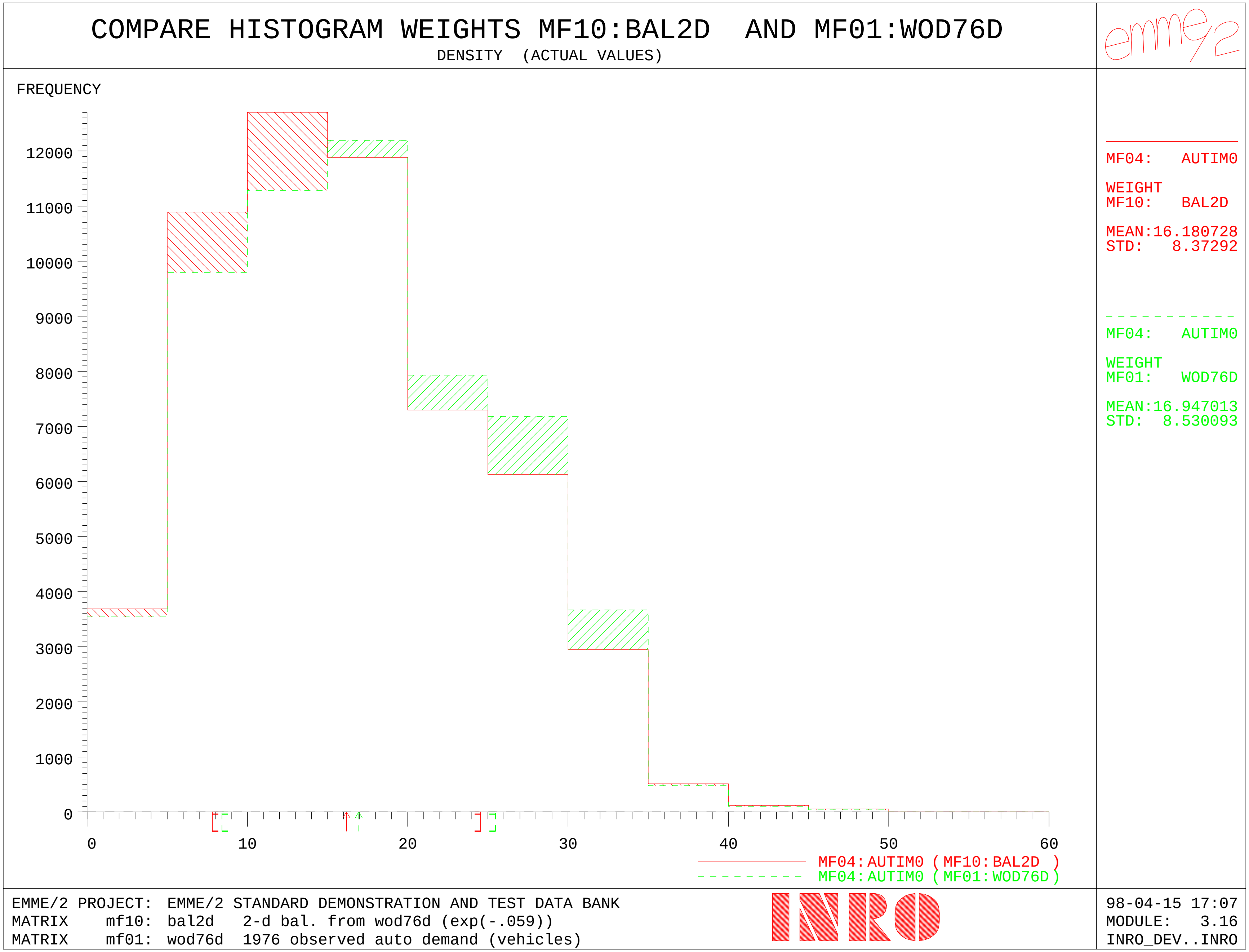
- Example:

```
~<balmprod md"attrac" mf"fwork" mo"pwork" mf"fstud" mo"pstud" mf"fshop" mo"pshop"
```

The resulting matrix for *shopping trips* can then be computed as

```
mo93*md91*mf"fshop"
```

Comparison of Travel Time Distribution (5-min Intervals)



Three-Dimensional Matrix Balancing (Module 3.22)

- Aim: determine an origin-destination matrix, g_{pq}, satisfying:

$$\sum_q g_{pq} = O_p \quad \text{for each } p \quad \textit{(productions are respected)}$$

$$\sum_p g_{pq} = D_q \quad \text{for each } q \quad \textit{(attractions are respected)}$$

$$g_{pq} \geq 0 \quad \text{for each } (p, q)$$

$$\sum_{(p,q)|k_{pq} = k} g_{pq} = F_k \text{ for each } k \quad \text{(total respected for each category)}$$

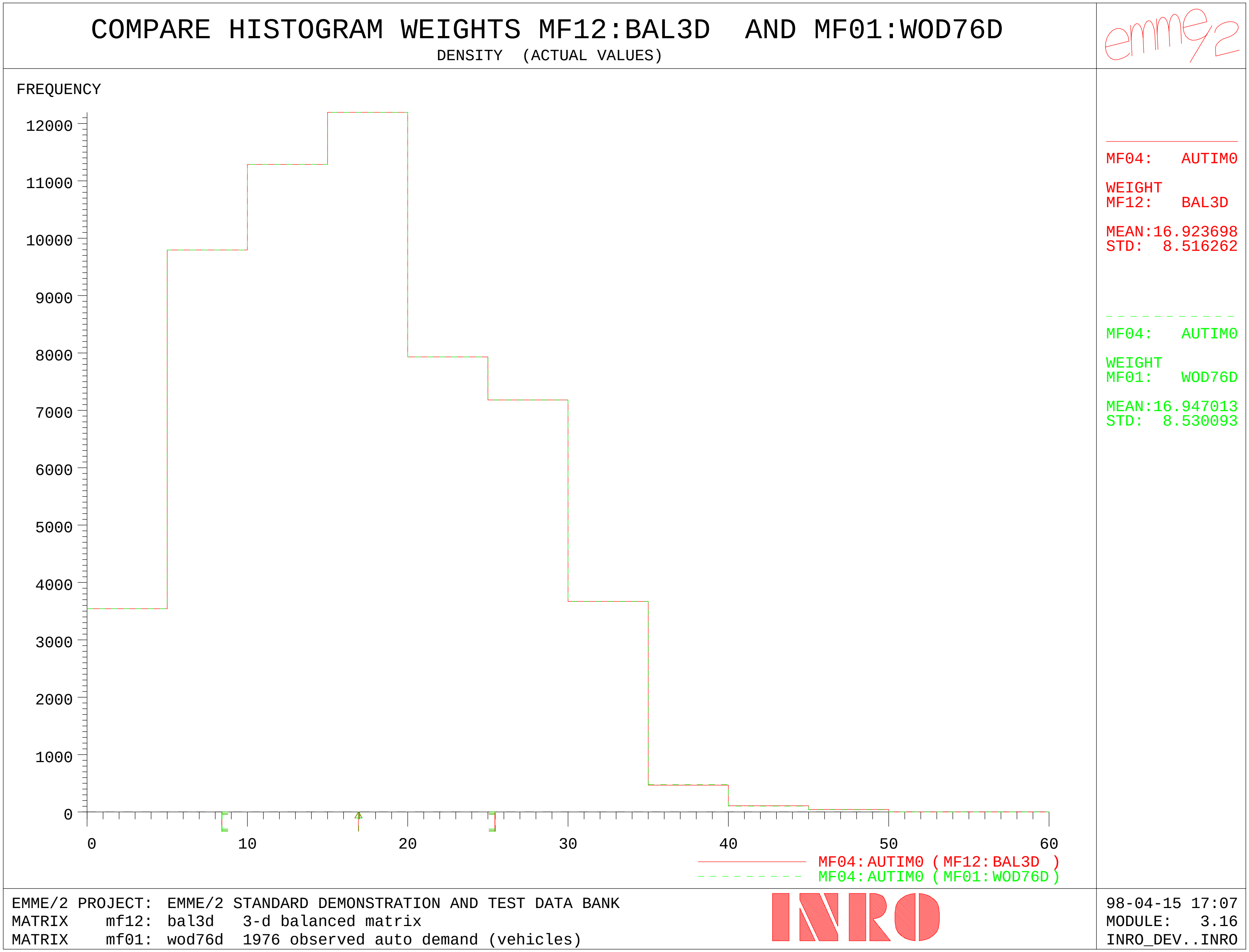
- Requires:
  - an index matrix specifying to which category each O-D pair belongs
  - totals for each **category** (district-to-district, travel time interval, …)

- Algorithm contains an additional step to balance the third dimension.

- The resulting **balanced matrix** can be expressed as $g_{pq} = \alpha_p \beta_q \gamma_{k_{pq}} f_{pq}$ where α_p β_q and $\gamma_{k_{pq}}$ are the **balancing coefficients.**

- Useful for **refining distribution models**, by including an additional constraint.

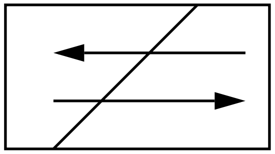
Comparison of Travel Time Distribution (5-min Intervals)



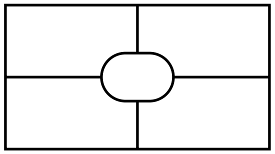
Refining a Trip Distribution Model

Include an additional constraint based on:

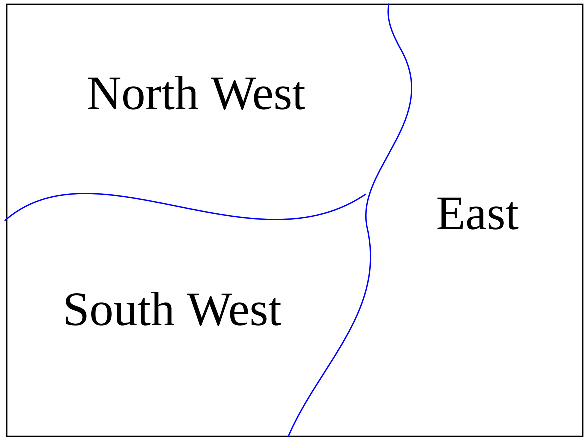
- Travel time distribution
- Screen lines
  - 3 categories of trips:
    trips crossing in each direction
    trips remaining on the same side
  - screen line counts must include all crossings



- Boundaries
  - physical and/or psychological (districts, rivers, etc.)
  - may make some minutes appear longer than others
  - define N sets of zones $\Rightarrow N^2$ categories of trips
  - set of totals must be reliable and consistent



Example: Reproducing Aggregated Totals



From	To	Observed	2-D model
NW	*NW*	16774	17694
NW	*E*	2530	2142
NW	*SW*	3513	2981
E	*NW*	9942	9499
E	*E*	6389	6741
E	*SW*	2921	3012
SW	*NW*	8402	7924
SW	*E*	1359	1393
SW	*SW*	4389	4830

Example: Reproducing Aggregated Totals (Continued)

- Number the categories. For example:

	NW	**E**	**SW**
NW	11	12	13
E	21	22	23
SW	31	32	33

- Compute the index matrix (module 3.21)

```
"index" = 10*gw(p)+gw(q)
```

- 3-dimensional balancing (module 3.22)
  - input matrix: **`"entrop"`**
  - 3-d index matrix: **`"index"`**
  - totals for third dimension: observed totals

- Aggregated totals for 3-D balanced matrix correspond exactly to those observed

```
origin destin observ  3dbal
  gw01   gw01  16774  16774
  gw01   gw02   2530   2530
  gw01   gw03   3513   3513
  gw02   gw01   9942   9942
  gw02   gw02   6389   6389
  gw02   gw03   2921   2921
  gw03   gw01   8402   8402
  gw03   gw02   1359   1359
  gw03   gw03   4389   4389
```

3-D Balancing with Group-to-Group Totals: BALANCE3

- Performs a 3-D matrix balancing using module 3.21 only
  ⇨ number of 3-D constraints not limited to a maximum of 250,
  but procedure less efficient than module 3.22.

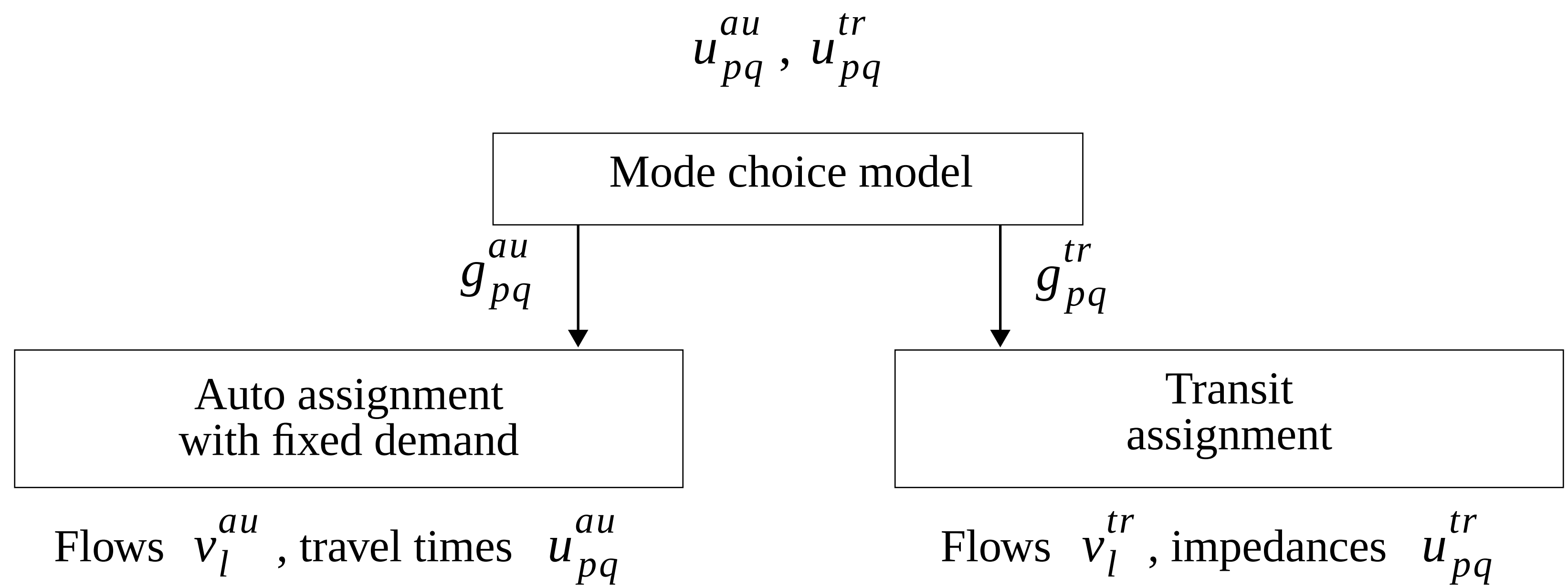
- 3-D constraints are the group-to-group totals of an aggregated matrix.

- Usage: **~<balance3** *Cpq Op Dq Rpq* **g***x* *Ap Bq Gpq*
  - *Cpq* contains the desired input matrix (entropy, etc.)
  - *Rpq* contains the 3rd dimension group-to-group totals (according to **g***x*) replicated in all corresponding O-D pairs

  Example:
  ~<balance3 "entrop" "produc" "attrac" "dtod" w mo97 md97 mf15

- The procedure might not converge if:
  - the group subtotals in matrix *Rpq* do not match the corresponding production and attraction subtotals in *Op* and *Dq*
  - …

Mode Choice Models



- Resulting u_{pq}^{au} , u_{pq}^{tr} not necessarily consistent with those used as input to mode choice model ⇒ various **feedback** techniques

- Many time and cost components needed for each O-D pair:
  - drive-alone: time, distance, toll, parking cost, …
  - car pool (HOV):time saving, …
  - transit with walk access: walk, in-vehicle, waiting, fare, …
  - transit with auto access: auto time, transit times, parking cost, …

Auto Assignment (Modules 5.11 & 5.21)

Standard assignment

- **Equilibrium** assignment (“all-or-nothing” assignment ⇒ iteration 0)
- Based on **time** or **generalized cost**
- Output matrix: *travel time* or *generalized cost*

Additional options assignment

- Computation of other **O-D attributes** such as *distance*, *toll*, …
- Typical set-up:

Additional attribute:	contents depends on desired result
Path operator:	**+**
Threshold:	none
O-D attribute:	path attribute

Transit Assignment (Modules 5.11 & 5.31)

Standard assignment

- **Multi-path** assignment based on *optimal strategies*
- Output matrices:
  - *transit time* (weighted sum)
  - *time components* (unweighted, for 1 or more mode)
  - *average number of boardings* (for 1 or more mode)

Additional options assignment

- Computation of other **O-D attributes** such as *distance*, *fare*, …
- Typical set-up:

Additional attributes:	which attribute and its contents depend on desired result
Path operator:	**+**
Sub-strategy comb. operator:	**`average`**
Retained part of strategies:	complete strategies
Threshold:	none
Matrix saved:	additional strategy attribute

Modelling Park-and-Ride: Best Total Time

- Aim:
  - for each O-D pair, find the "best" parking lot to use for P&R, that is, the one for which $a_{pk} + t_{kq} + c_k$ is minimum
  - obtain **corresponding travel time matrices** for car and transit

- Required data:
  - network where parking lots are coded as centroids (⇨ zones)
  - auto time (or impedance) matrix
  - transit time (or impedance) matrix
  - parking cost vector (optional)

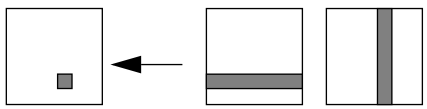
- Desired results
  - matrix containing the **zone number of the parking lot** with

$$\text{Min} \quad (a_{pk} + t_{kq} + c_k)$$

  - two **impedance matrices**:
    - **auto** impedance from origin to best parking lot
    - **transit** impedance from best parking lot to destination

Matrix Convolutions (Module 3.23)

- Matrix product like operations
- Each element r_{pq} of result matrix is obtained from the elements of row p of the 1st operand matrix and column q of the 2nd operand matrix



- Useful for **calculations involving intermediate zones**, including more complex demand models

From Matrix Product to Matrix Convolution

Matrix product

- Each element of row p of A is **multiplied** by the corresponding element of column q of B, and the intermediate results are **summed**
- $r_{pq} = \sum_k a_{pk} \times b_{kq}$

Triple-index operation

- Different combination $\otimes$ and contraction $\oplus$ operators may be used
- $r_{pq} = \bigoplus_k a_{pk} \otimes b_{kq}$

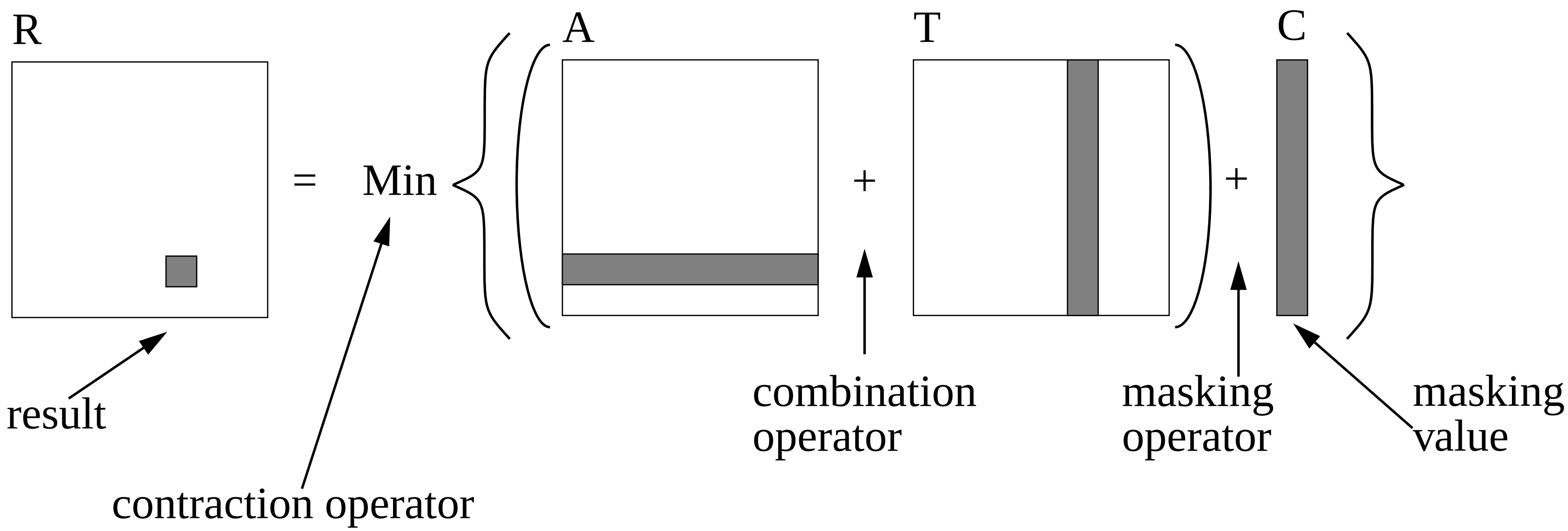
Convolution

- **Further processing** may be performed on $a_{pk} \otimes b_{kq}$, before contraction operator $\oplus$ is applied to the intermediate results

Finding the Best Parking Lot (Convolution)

Minimum total time:

$$r_{pq} = \min_k \{(a_{pk} + t_{kq}) + c_k\}$$



Corresponding parking lot: $l_{pq} = \underset{k}{\mathrm{Argmin}} \{(a_{pk} + t_{kq}) + c_k\}$

Extracting the Auto and Transit Travel Times

- Aim: **extract the impedance for each leg** of the trips
  Upk for leg 1
  Ukq for leg 2
  based on the *selected parking lot*

- Macro: **`~<legimped`** *leg Upq Kpq Lpq*
  leg 1 or 2
  Upq impedance matrix to extract leg impedances from
  Kpq index matrix containing intermediate zone numbers
  Lpq resulting first or second leg impedance matrix

  Example:

```
~<legimped 1 autim0 bstlot mf11
~<legimped 2 trtim0 bstlot mf12
```

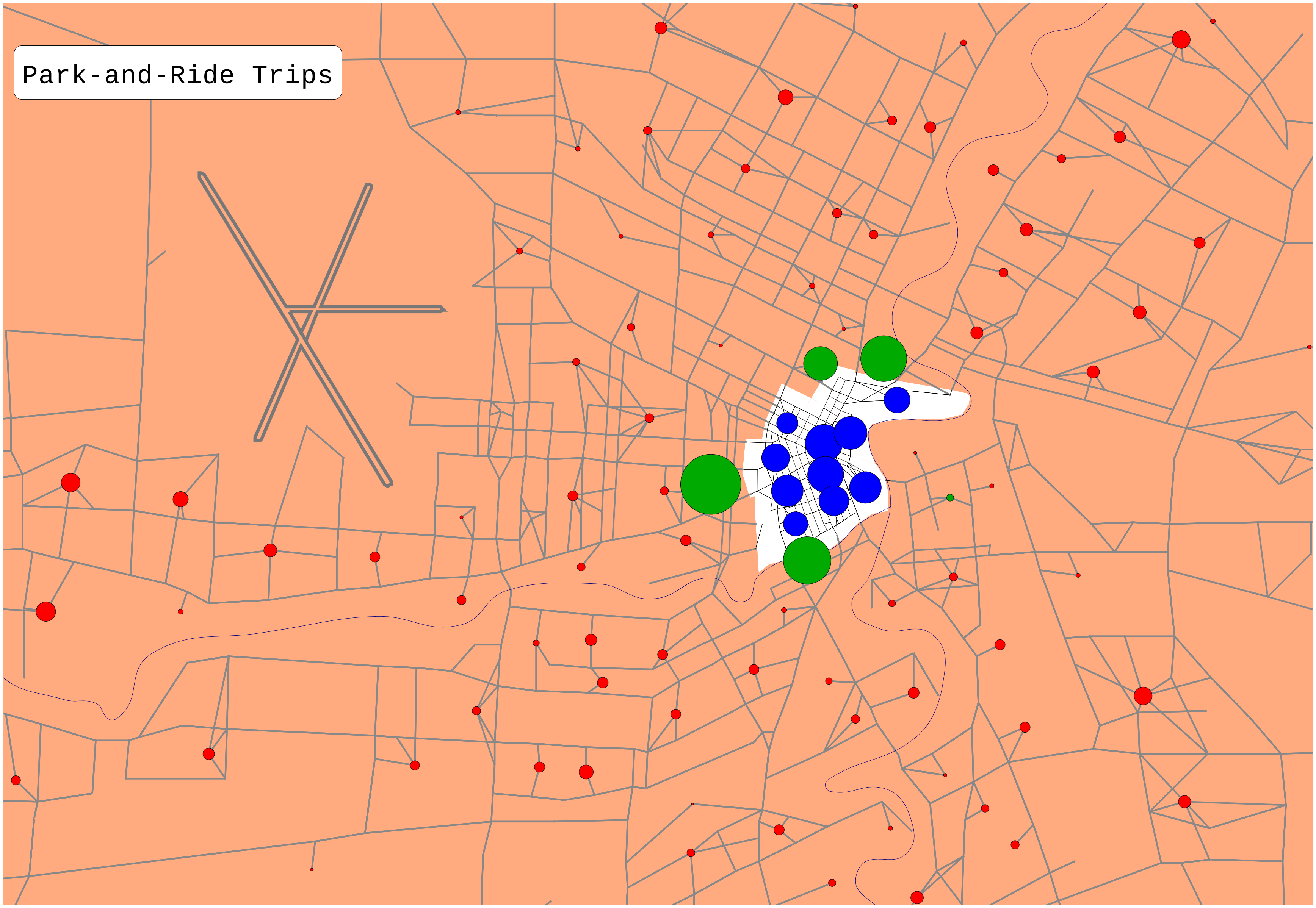
Splitting the P&R Demand into Auto and Transit Demand

- Aim: **split the park-and-ride demand for the 2 legs** of the trips
  Gpk for leg 1
  Gkq for leg 2
  based on the *selected parking lot*

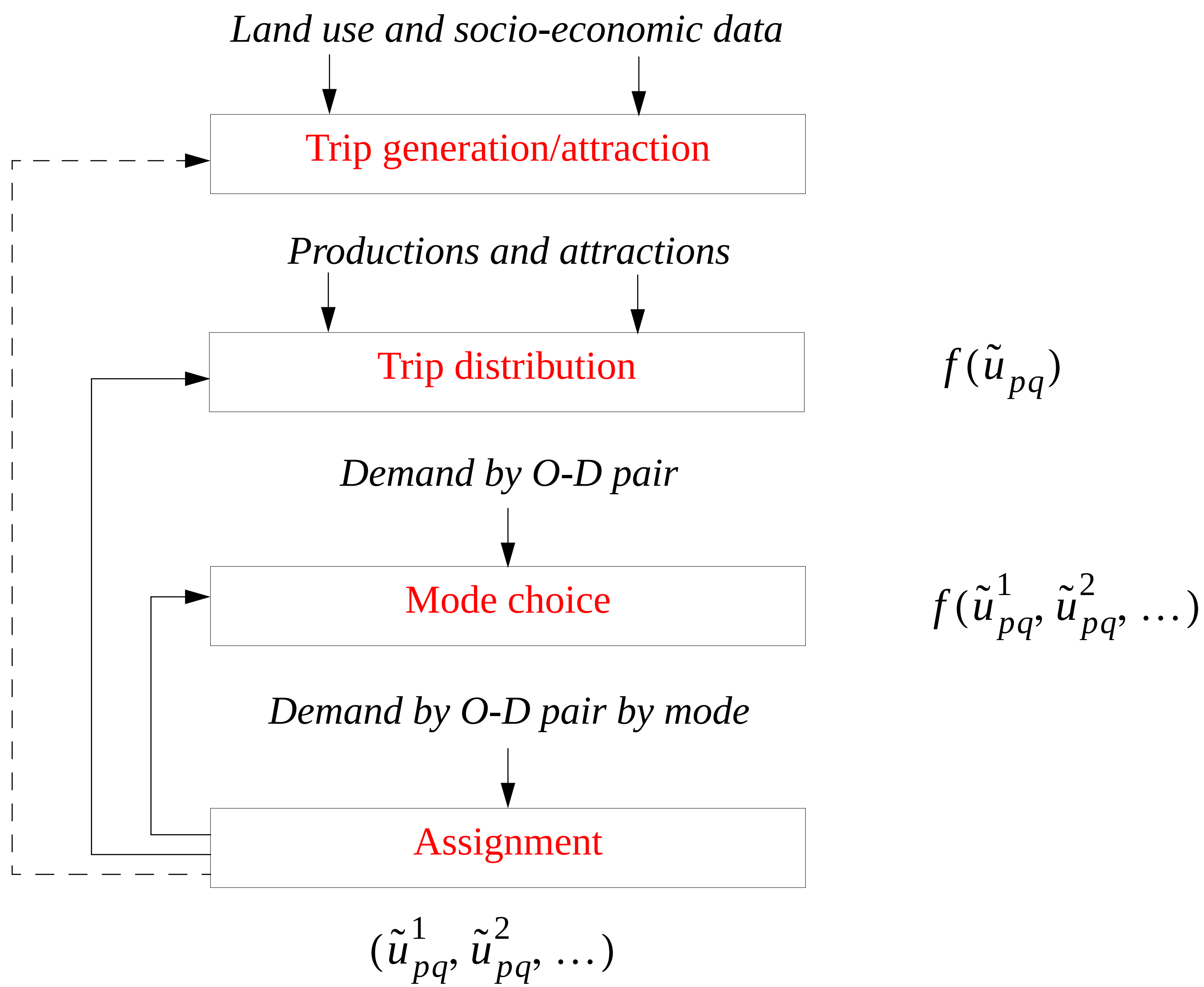
- Macro: **`~<splitmat`** *Gpq Kpq Gpk Gkq*
  Gpq demand matrix to be split
  Kpq index matrix containing intermediate zone numbers
  Gpk demand from the origins to the intermediate zones
  Gkq demand from the intermediate zones to the destinations

  Example:
  `~<splitmat prdem prlot mf9 mf10`

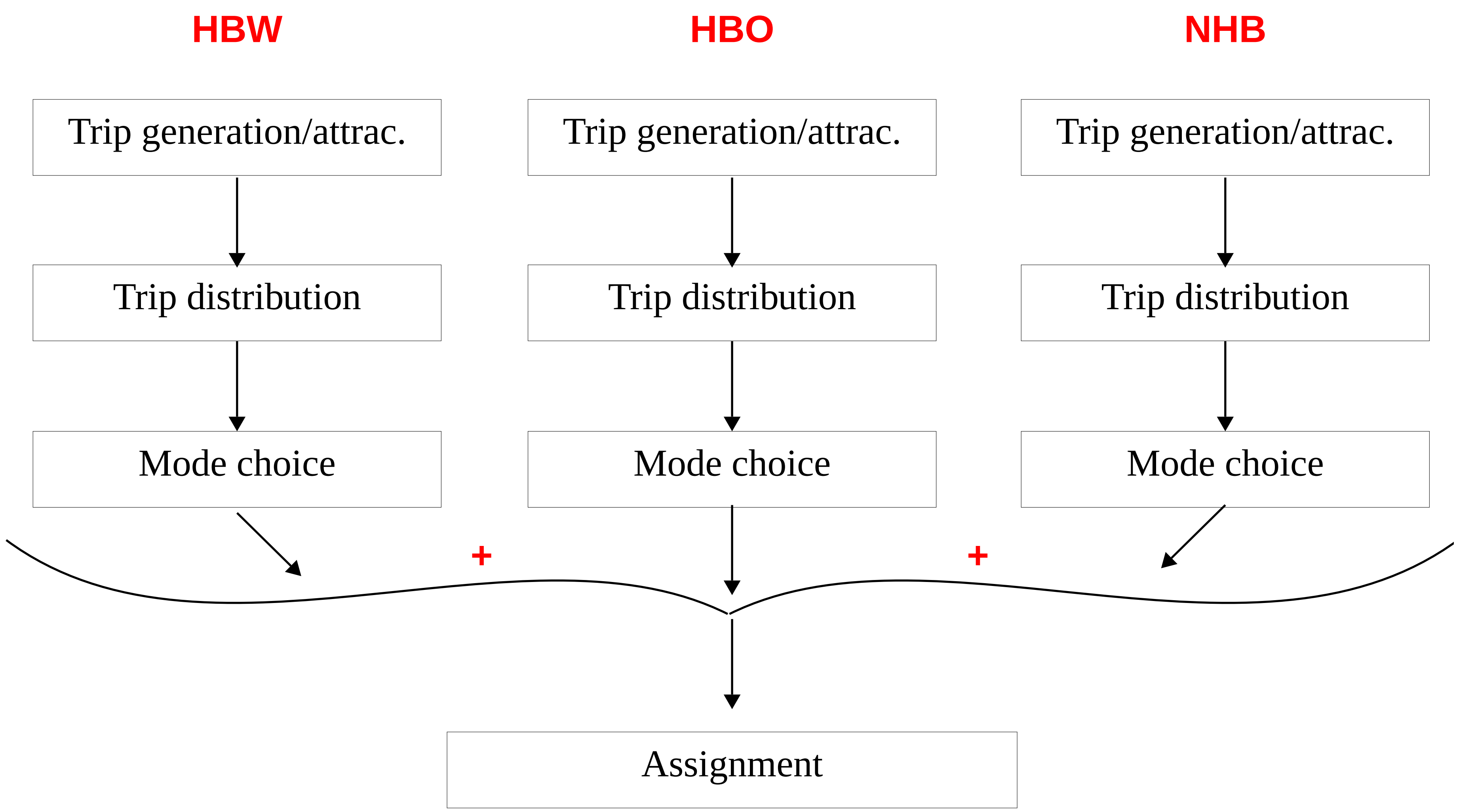
Park-and-Ride Trips



Feedback in a 4-Step Model



4-Step Model with Multiple Trip Purposes



Conclusions

Flexible framework for demand modelling!

⇒ matrix manipulation tools

⇒ assignment procedures

⇒ macros available on the web
www.spiess.ch

Share experience!

⇒ papers available on the web
www.inro.ca

⇒ list server for EMME/2 users
www.egroups.com/group/emme2users